

Lecture 1:

Introduction to Machine Learning

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, Jessica Wu (Harvey Mudd), David Kauchak (Pomona), Dan Roth (Upenn), Sriram Sankararaman (UCLA), whose slides are also heavily used, and the many others who made their course materials freely available online.

Registration

- ❖ Please fill the following form for PTE:
<https://bit.ly/cm146f22-signup>
 - ❖ Limited by the resources
 - ❖ Unlikely we can give many PTEs
- ❖ Please drop the course if you're not planning to take it
- ❖ PTE will be given on the second week
 - ❖ Check <http://my.ucla.edu>

CS M146 Teaching Team

- ❖ Kai-Wei Chang
 - ❖ Wed, 12:00 PM – 1:00 PM
- ❖ TAs

TA	Email	Office Hours
Fan Yin	fanyin20@cs.ucla.edu	Thu 3-5pm
Zhouxing Shi	zshi@cs.ucla.edu	Wed 8:30pm-9:30pm Thu 2-3pm
Tanmay Parekh	tparekh@g.ucla.edu	Mon 1-3pm
Yihe Deng	yihedeng@ucla.edu	Wed 11am-1pm
Sidi Lu	sidi.lu@cs.ucla.edu	Wed 10:30am - 11:30am, Friday 10:30am - 11:30am
Masoud Monajatipoor	monajati@ucla.edu	

Homework 0

- ❖ Sign up at Piazza
<http://piazza.com/ucla/fall2022/m146>
- ❖ Complete the math quiz at Bruinlearn
 - ❖ Will be released on Friday

What is machine learning?

Machine Learning is the study of algorithms that

- improve their performance P
- at some task T
- with experience E .

A well-defined learning task is given by $\langle P, T, E \rangle$.

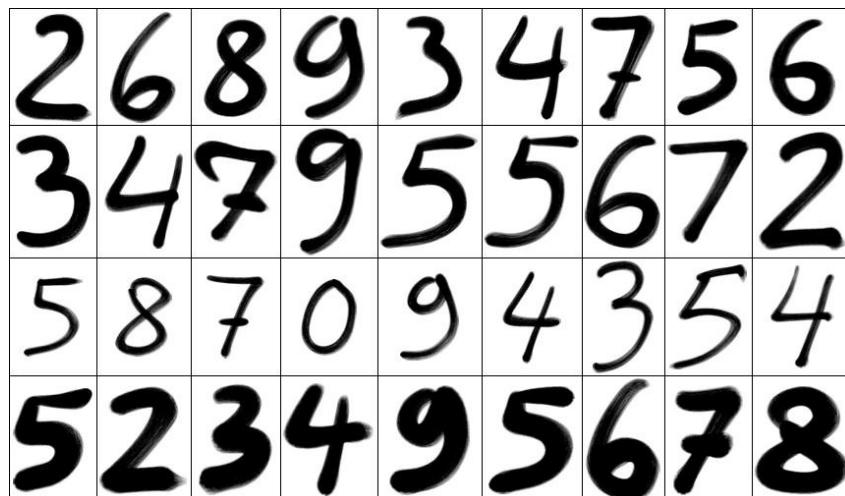
[Definition by Tom Mitchell (1998)]

Improve on task T with respect to performance P, based on experience E

T: Recognizing hand-written words

P: Percentage of words correctly classified

E: Database of human-labeled images of hand written words

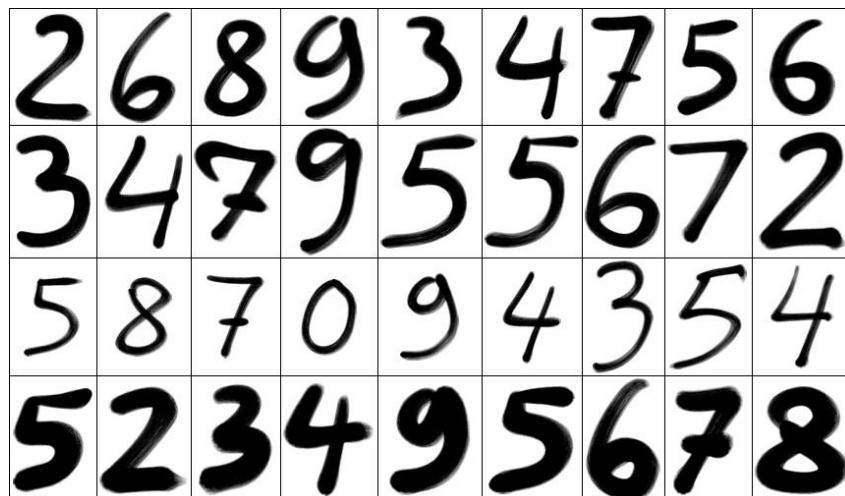


Improve on task T with respect to performance P, based on experience E

T: Recognizing hand-written words

P: Percentage of words correctly classified

E: Database of human-labeled images of hand written words



Improve on task T with respect to performance P, based on experience E



Improve on task T with respect to performance P, based on experience E

❖ GPT-X language models

<https://huggingface.co/gpt2>

Text Generation Examples ▾

UCLA is the best university. In fact, it is the top-ranked academic community in the US, and is the only community in its state that makes the top ten lists of most expensive research institutions. UCLA is known for it's competitive sports|

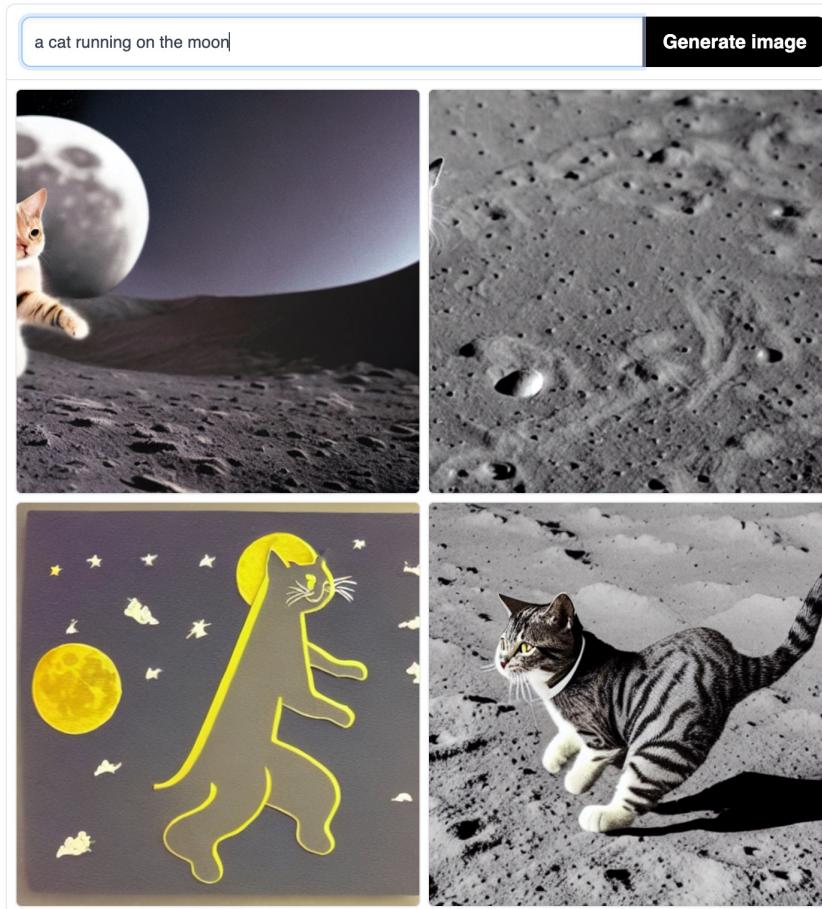
Compute⌘+Enter 1.5

2+

Improve on task T with respect to performance P, based on experience E

❖ Stable Diffusion

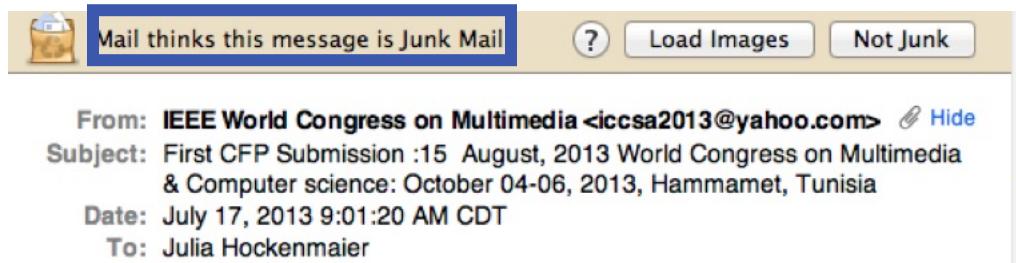
<https://huggingface.co/spaces/stabilityai/stable-diffusion>



Discussion

- ❖ [2 min] Introduce yourself to your group
 - ❖ Your name, your major and one interesting fact
- ❖ [5 min] Brainstorm:
What is your dream application of ML?
 - ❖ Define (Task, Performance, Experience)
 - ❖ Can be something not exist
(e.g., a robot writing homework for you)
- ❖ [3 min] Pick the best answer and the presenter
and post the presenter's name at chat box

Applications: Spam Detection



- ❖ This is a **binary classification task**:
Assign one of two labels (i.e. yes/no) to the input (here, an email message)
- ❖ Classification requires a model (a classifier) to determine which label to assign to items.
- ❖ In this class, we study algorithms and techniques to learn such models from data.

The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition (e.g., vision)



The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition
 - ❖ Perform knowledge intensive inferences



The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition
 - ❖ Perform knowledge intensive inferences
 - ❖ Deal with messy, real world data



Learning = Generalization

H. Simon (Turing Award 1975, Nobel Prize 1978)-

“Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the task or tasks drawn from the same population more efficiently and more effectively the next time.”

The ability to perform a task in a situation which has never been encountered before

Learning = Generalization



Mail thinks this message is junk mail.

Not junk

- ❖ The learner has to be able to **classify** items it has never seen before.

Administrivia

Prerequisites

- ❖ The pillars of machine learning
 - ❖ Probability and statistics
 - ❖ Linear algebra
 - ❖ Calculus/Optimization

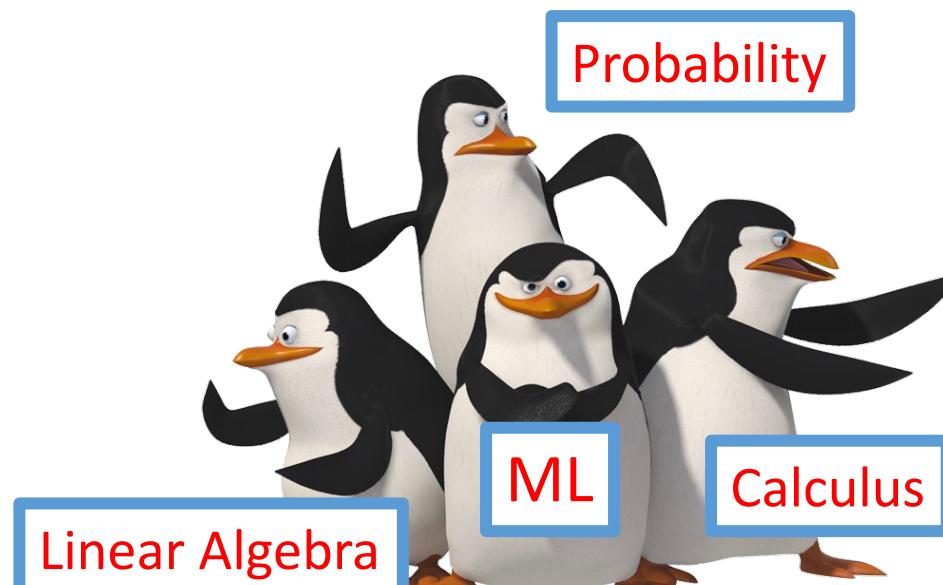


Image adapted from <http://pngimg.com/download/31173>
under CC by-NC 4.0

Prerequisites

- ❖ The pillars of machine learning
 - ❖ Probability and statistics
 - ❖ Linear algebra
 - ❖ Calculus/Optimization
 - ❖ Computer science background
 - ❖ Algorithms
 - ❖ Programming experience
- We will use **Python and scikit-learn**

In-Person Lecture

- ❖ Your participation is appreciated
- ❖ Questions are welcomed
- ❖ Audio/video recording may be available at BruinLearn
 - ❖ You should not rely on these recordings as a substitute for lectures
- ❖ Although not a formal component of the grade, attendance is important

Problem Set

- ❖ Problem Sets
 - ❖ Three problem sets
 - ❖ Due at 11:59pm on the due date
 - ❖ 24hr late credits for the entire quarter
 - ❖ Will be using GradeScope to manage submissions
(submission instructions will be provided in the discussion session)
- ❖ All solutions must be clearly written or typed.
 - ❖ Unreadable answers will not be graded. We encourage using LaTeX to type answers.
 - ❖ Solutions will be graded on both correctness and clarity
- ❖ For programming HW, upload your source code at Bruinlearn

Exams

- ❖ Midterm is a **3hr online open-book exam**
- ❖ Final is a **3hr in-person closed-book exam on paper**
- ❖ Exam will cover materials from the lectures and the problem sets.
- ❖ No alternate or make-up exams
 - ❖ Except for disability/medical/emergency reasons documented and communicated to the instructor prior to the exam date.
 - ❖ Exam date and time **cannot** be changed to accommodate scheduling conflicts with other classes or job fair/interview.

Regrading request

- ❖ Must be made **within one week** after the grade is released regardless of any reason
- ❖ We reserve the right to regrade entire problem set for a given regrade request.

Quiz

- ❖ We will have quizzes (almost) every week after week 2
- ❖ A handful multiple choice questions
- ❖ You have only one try
- ❖ One lowest quiz score will be dropped

Final grade



- ❖ Default cut-off for letter grade is:

> 97	93	90	87	83	80	77	73	70	< 70
A +	A	A -	B +	B	B -	C +	C	C -	D

- ❖ We **will not** make adjustments for individuals
 - ❖ E.g., no round up (i.e., 89.99 = B+)
- ❖ In general, we **will not** curve the final grades, but may do some adjustment if needed
 - ❖ The cut-off score will only get lower (i.e., you may get a better letter grade)
- ❖ This is a **heavy** course
- ❖ Score distribution may be different from last year

Academic integrity policy

- ❖ **No cheating**
 - ❖ In particular, you are free to discuss homework problems. However, you **must** write up your own solutions (solution/program). You **must** also acknowledge all collaborators.
 - ❖ Please don't use any old solution you found.
 - ❖ All incidents will report to the student office
- ❖ **Don't post your HW/Exam solutions or upload course materials without consent**

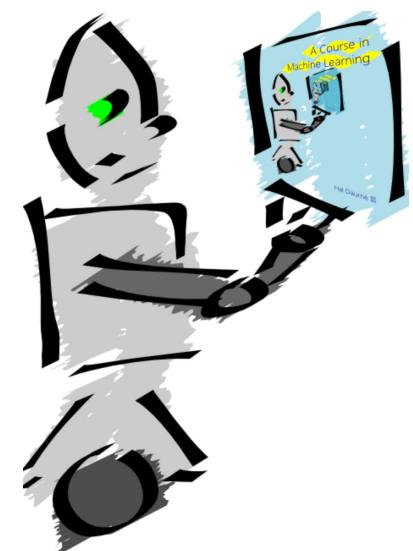
CM146 on Web



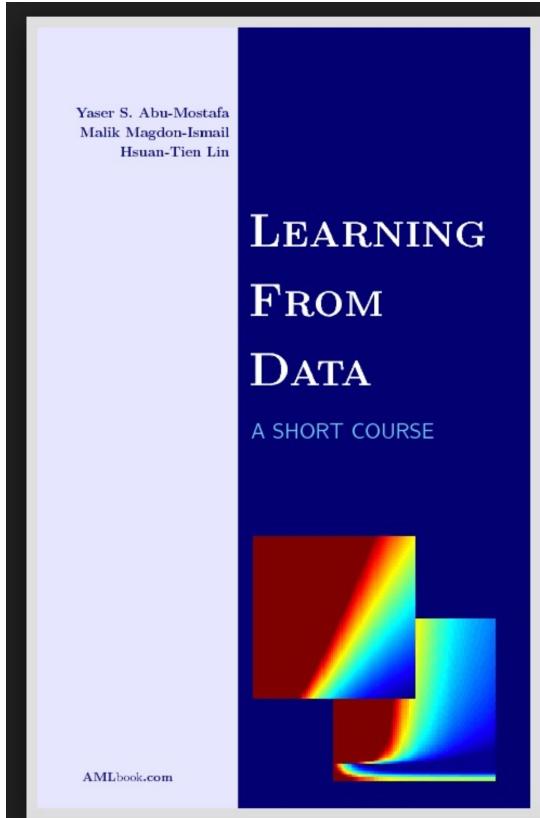
- ❖ Course website:
<https://ccle.ucla.edu/course/view/21F-COMSCIM146-1>
- ❖ Piazza:
<http://piazza.com/ucla/fall2022/m146>
 - ❖ Strongly encourage students to post here (publicly or privately) rather than email staff directly (you will get a faster response this way)
- ❖ Gradescope
 - ❖ Maintain homework/final
 - ❖ Access through Bruinlearn

Textbook

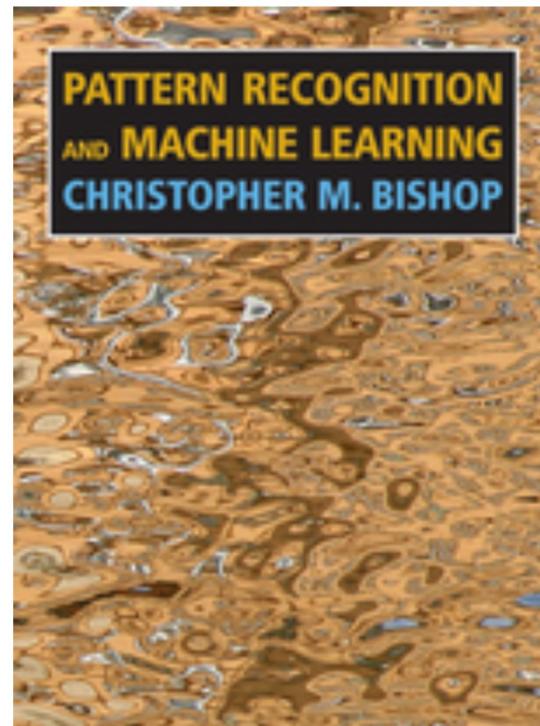
- ❖ No textbook
 - ❖ Primary reference:
A course in machine learning by Hal Daume III
(CIML). Freely available online <http://ciml.info/>
 - ❖ See syllabus for the reading list



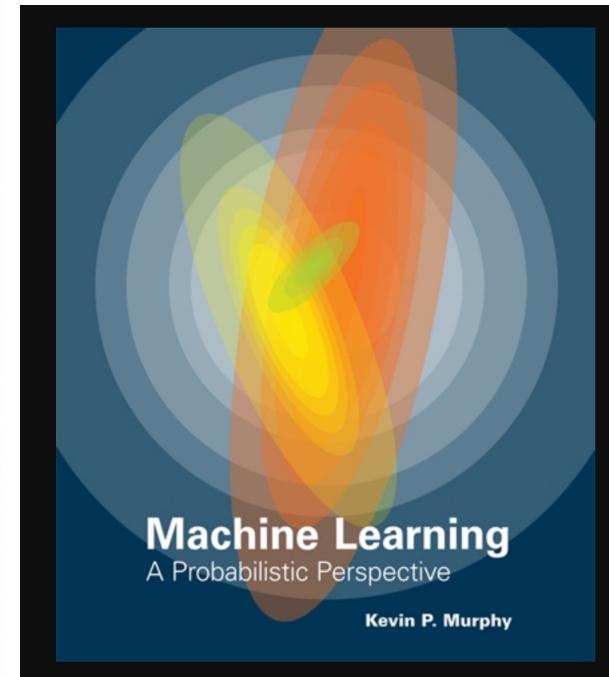
Other references



Basic



Comprehensive



Advanced

Why taking this course?

Building fundamental knowledge

A Regularized Framework for Sparse and Structured Neural Attention

Vlad Niculae*
Cornell University
Ithaca, NY
vlad@cs.cornell.edu

Mathieu Blondel
NTT Communication Science Laboratories
Kyoto, Japan
mathieu@mblondel.org

Abstract

Modern neural networks are often augmented with an attention mechanism, which tells the network where to focus within the input. We propose in this paper a new framework for sparse and structured attention, building upon a smoothed max operator. We show that the gradient of this operator defines a mapping from real values to probabilities, suitable as an attention mechanism. Our framework includes softmax and a slight generalization of the recently-proposed sparsemax as special cases. However, we also show how our framework can incorporate modern structured penalties, resulting in more interpretable attention mechanisms, that focus on entire segments or groups of an input. We derive efficient algorithms to compute the forward and backward passes of our attention mechanisms, enabling their use in a neural network trained with backpropagation. To showcase their potential as a drop-in replacement for existing ones, we evaluate our attention mechanisms on three large-scale tasks: textual entailment, machine translation, and sentence summarization. Our attention mechanisms improve interpretability without sacrificing performance; notably, on textual entailment and summarization, we outperform the standard attention mechanisms based on softmax and sparsemax.

1 Introduction

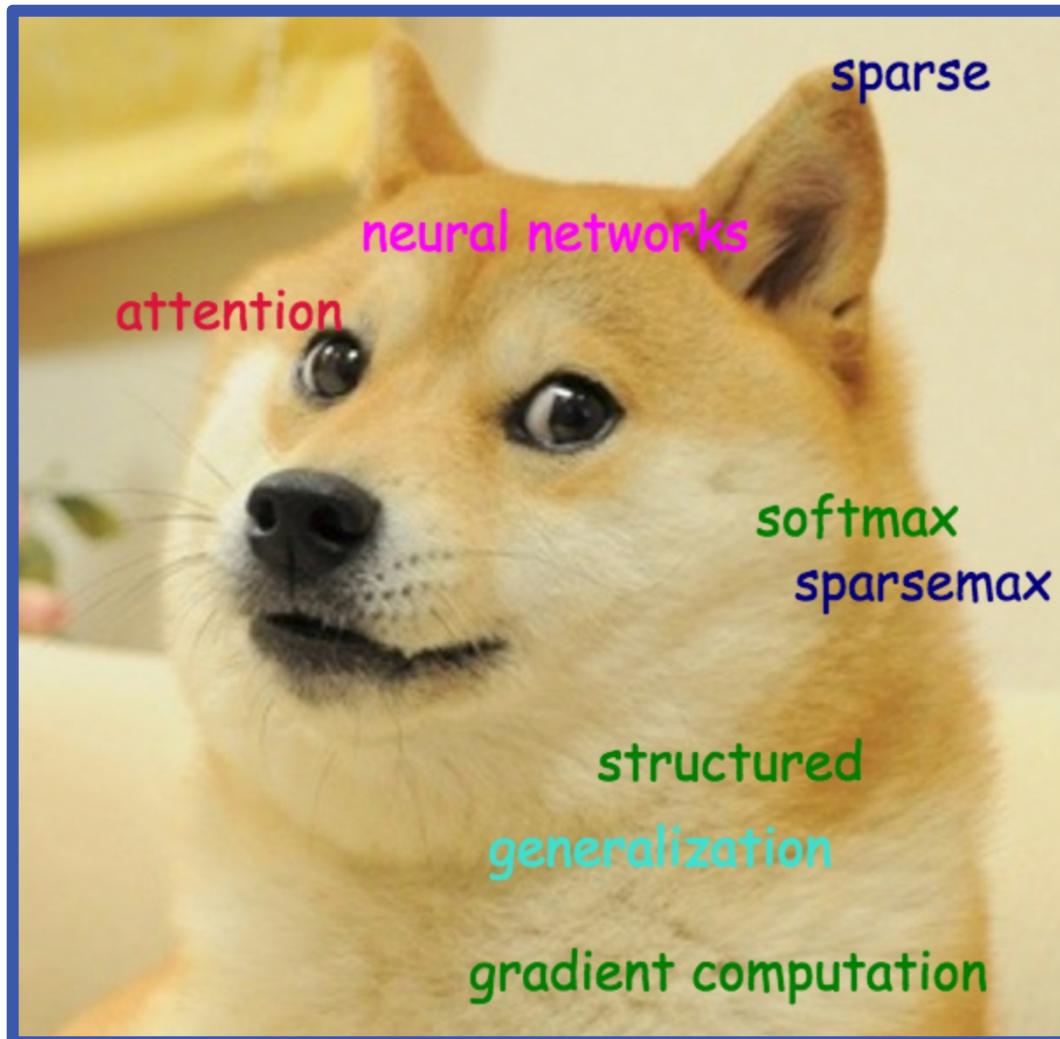
Modern neural network architectures are commonly augmented with an attention mechanism, which tells the network where to look within the input in order to make the next prediction. Attention-augmented architectures have been successfully applied to machine translation [2, 29], speech recognition [10], image caption generation [44], textual entailment [38, 31], and sentence summarization [39], to name but a few examples. At the heart of attention mechanisms is a mapping function that converts real values to probabilities, encoding the relative importance of elements in the input. For the case of sequence-to-sequence prediction, at each time step of generating the output sequence, attention probabilities are produced, conditioned on the current state of a decoder network. They are then used to aggregate an input representation (a variable-length list of vectors) into a single vector, which is relevant for the current time step. That vector is finally fed into the decoder network to produce the next element in the output sequence. This process is repeated until the end-of-sequence symbol is generated. Importantly, such architectures can be trained end-to-end using backpropagation.

Alongside empirical successes, neural attention—while not necessarily correlated with human attention—is increasingly crucial in bringing more **interpretability** to neural networks by helping explain how individual input elements contribute to the model’s decisions. However, the most commonly used attention mechanism, *softmax*, yields dense attention weights: all elements in the input always make at least a small contribution to the decision. To overcome this limitation, *sparsemax* was recently proposed [31], using the Euclidean projection onto the simplex as a sparse alternative to

Modern **neural networks** **attention mechanism**, ... We propose in this paper a new framework for **sparse** and **structured** attention, building upon a **smoothed max operator**. We show that the **gradient** of this operator defines a mapping from real values to **probabilities**, suitable as an attention mechanism. Our framework includes **softmax** and a slight **generalization** of the recently-proposed **sparsemax** as **special cases**.

*Work performed during an internship at NTT Communication Science Laboratories, Kyoto, Japan.

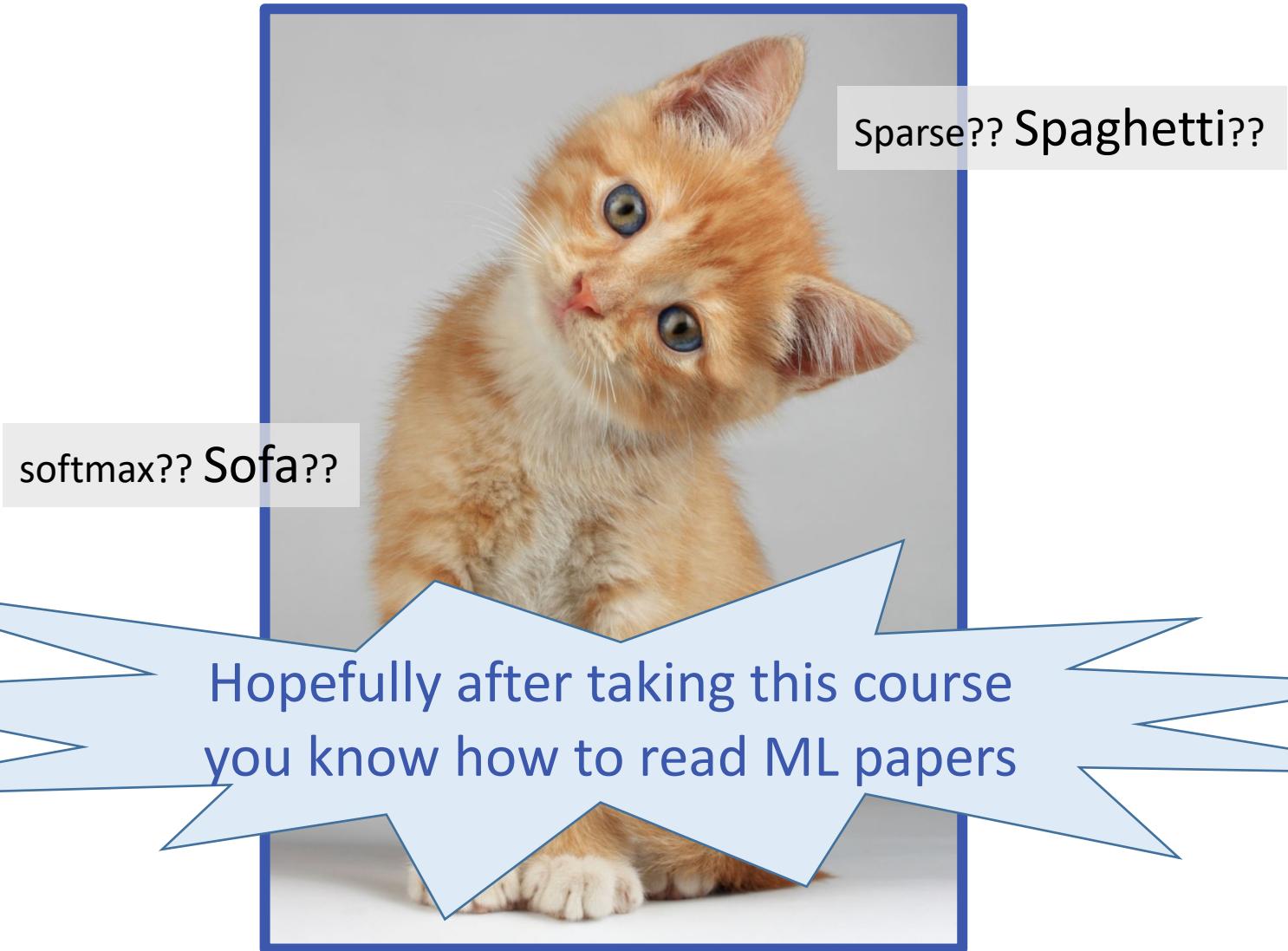
What it looks like to ML researchers



What it looks like to normal people



What it looks like to normal people



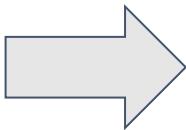
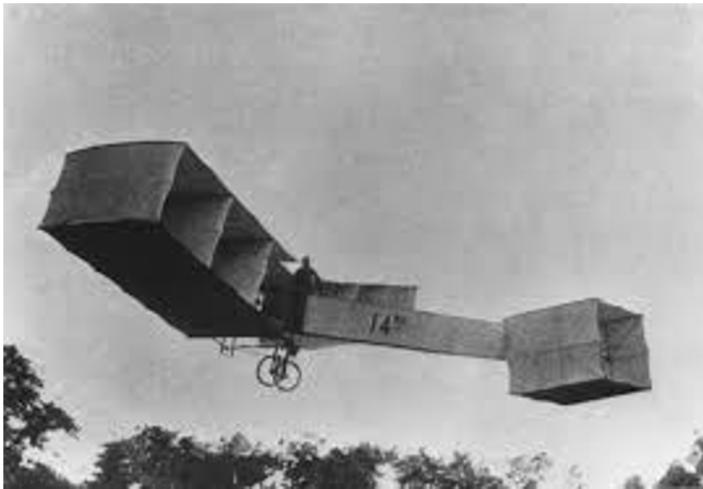
Goals of this course: Learn about

- ❖ Fundamental concepts and algorithms
 - ❖ Customize your own algorithm
- ❖ Common techniques/tools used
 - ❖ theoretical understanding
 - ❖ practical implementation
 - ❖ best practices
- ❖ How to "debug" ML system
- ❖ Black magic => systematic process

Why Study Machine Learning Now?

- ❖ Exciting moments for ML:
 - ❖ Initial **algorithms** and **theory** in place.
 - ❖ Growing amounts of on-line data
 - ❖ Computational power available.

Current Status



Current status:

- Compelling results on benchmarks,
- Work well in general domain
- Commercial uses

Challenges:

- Incorporate w/ human knowledge
- Reliable (fair, robust, interpretable) models that earn human trust
- Applications in specific domains
- Skewness of available annotation data

What will we learn?

- ❖ Supervised learning
 - ❖ Decision tree, Perceptron, Linear models, support vector machines, kernel methods, probabilistic models
- ❖ Unsupervised learning
 - ❖ Clustering,
 - ❖ EM algorithms
- ❖ Learning theory
- ❖ Deep learning (representation learning)
- ❖ Practical Issues
 - ❖ Experimental evaluation; Implementing ML models

Machine Learning is Interdisciplinary

- ❖ Makes Use of:
 - ❖ Probability and Statistics; Linear Algebra; Calculus; Theory of Computation;
- ❖ Related to:
 - ❖ Philosophy, Psychology ,Neurobiology, Linguistics, Vision, Robotics,....
- ❖ Has applications in:
 - ❖ AI (Natural Language; Vision; Planning; HCI)
 - ❖ Engineering (Agriculture; Civil; ...)
 - ❖ Computer Science (Compilers; Architecture; Systems; data bases...)

Other Related Courses

- ❖ CS145 - Introduction to Data Mining
- ❖ CM148 - Introduction to Data Science
- ❖ CS161 - Fundamentals of Artificial Intelligence
- ❖ Computer Vision/Bioinformatic/NLP
- ❖ Related course at ECE, Stats, Math dep.
- ❖ Graduate-level courses

Challenges in ML

Structured Inference

- ❖ Many predictions are compositional
 - ❖ Require an inference process

Structured Inference



Structured Inference



小心:
Carefully
Careful
Take
Care
Caution



地滑:
Slide
Landslip
Wet Floor
Smooth

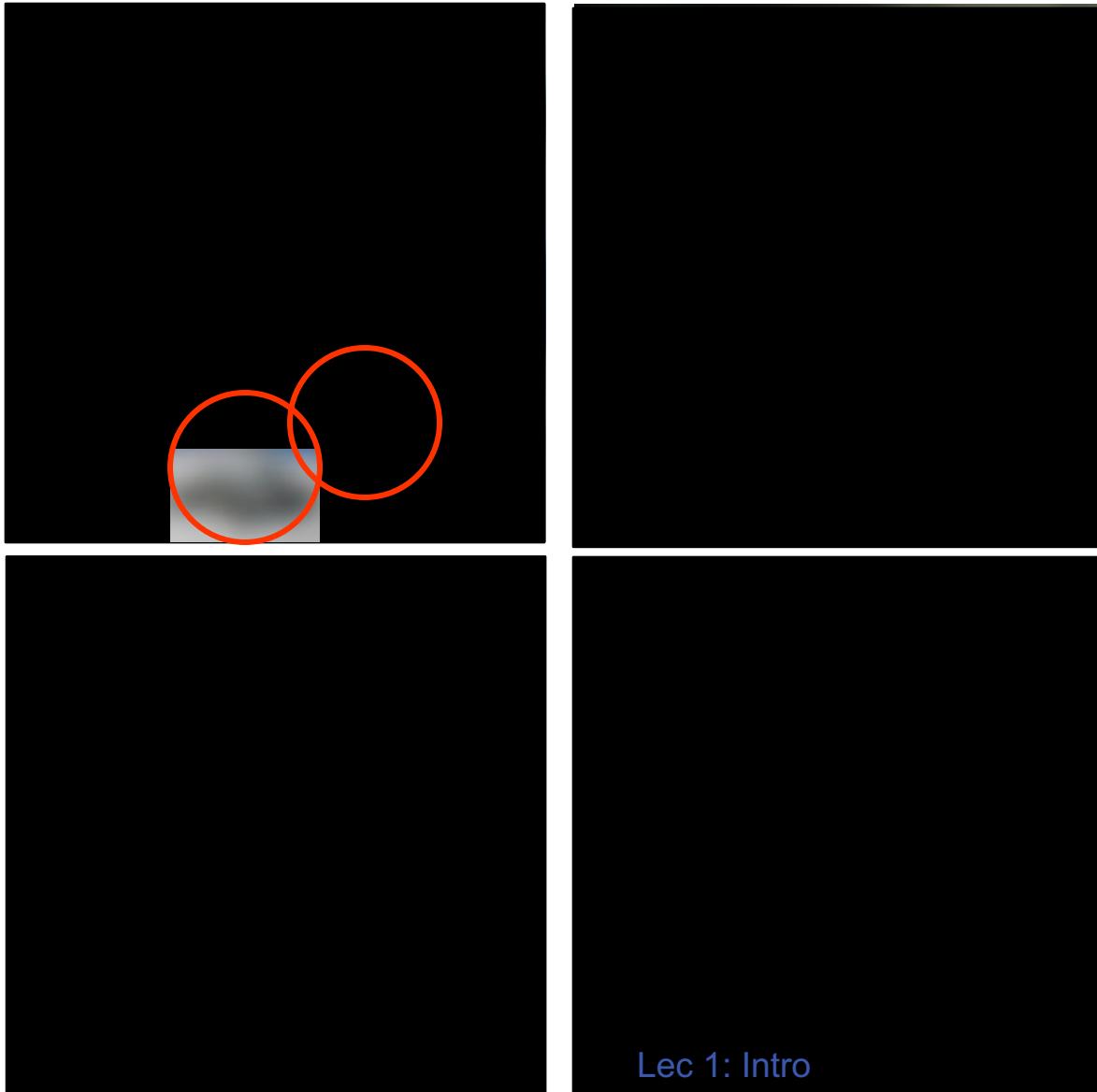
Translate

English Spanish French Chinese - detected English Spanish Arabic Translate

小心地滑 Carefully slide

Xǐngxīn dì huá

Structured Inference



Credit: Dhruv Batra

Robustness

Car or shoe?



TensionNotes.com



Adversarial Attack



93%, 20 Km/h Sign

+ ϵx



$sign(\nabla * J(\theta, x, y))$

=



90%, 80 Km/h Sign



<https://arxiv.org/abs/1712.09327v1>

Fairness in ML

Select photo  

X The photo you want to upload does not meet our criteria because:

- Subject eyes are closed

Please refer to the technical requirements.
You have 9 attempts left.

Check the photo [requirements](#).

[Read more about common photo problems and](#)

Subject eyes are closed

start again and re-enter the CAPTCHA security check.

Reference number: 20161206-81

Filename: Untitled.jpg

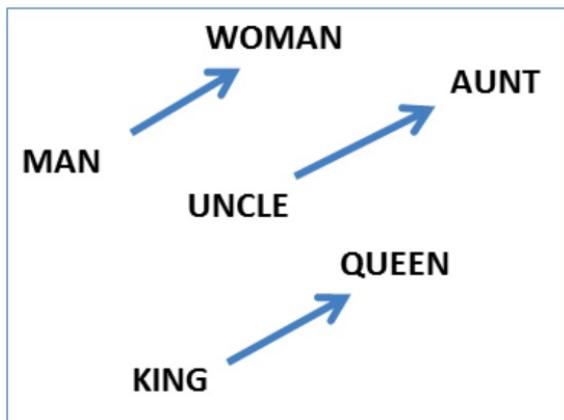
If you wish to [contact us](#) about the photo, you must provide us with the reference number given above.

Please print this information for your records.



Fairness in ML-- Word embedding bias

❖ $v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$

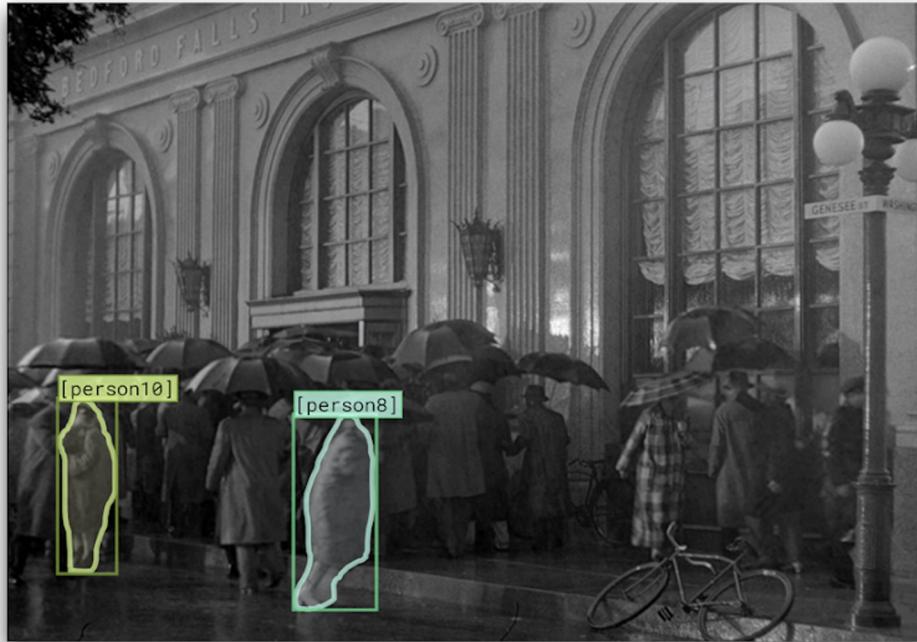


he: __	she: __
uncle	aunt
lion	
surgeon	
architect	
beer	
professor	



We use Google w2v embedding trained from the news

Commonsense



Is it raining outside?

- a) Yes, it is snowing.
- b) Yes, [person8] and [person10] are outside.
- c) No, it looks to be fall.
- d) Yes, it is raining heavily.

An example from the VCR dataset

Lecture 2:

Overview Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, Jessica Wu (Harvey Mudd), David Kauchak (Pomona), Dan Roth (Upenn), Sriram Sankararaman (UCLA), whose slides are also heavily used, and the many others who made their course materials freely available online.

Announcement

- ❖ There is a discussion session on Friday
 - ❖ See session/time/loc at myUCLA
- ❖ Math Review Quiz is on BruinLearn

This Lecture

- ❖ Learning Protocols
 - ❖ Supervised Learning
 - ❖ Unsupervised Learning
- ❖ Challenges in ML
- ❖ Framing Learning Problems

Type of learning protocols

Supervised Learning

Training phase:



lion



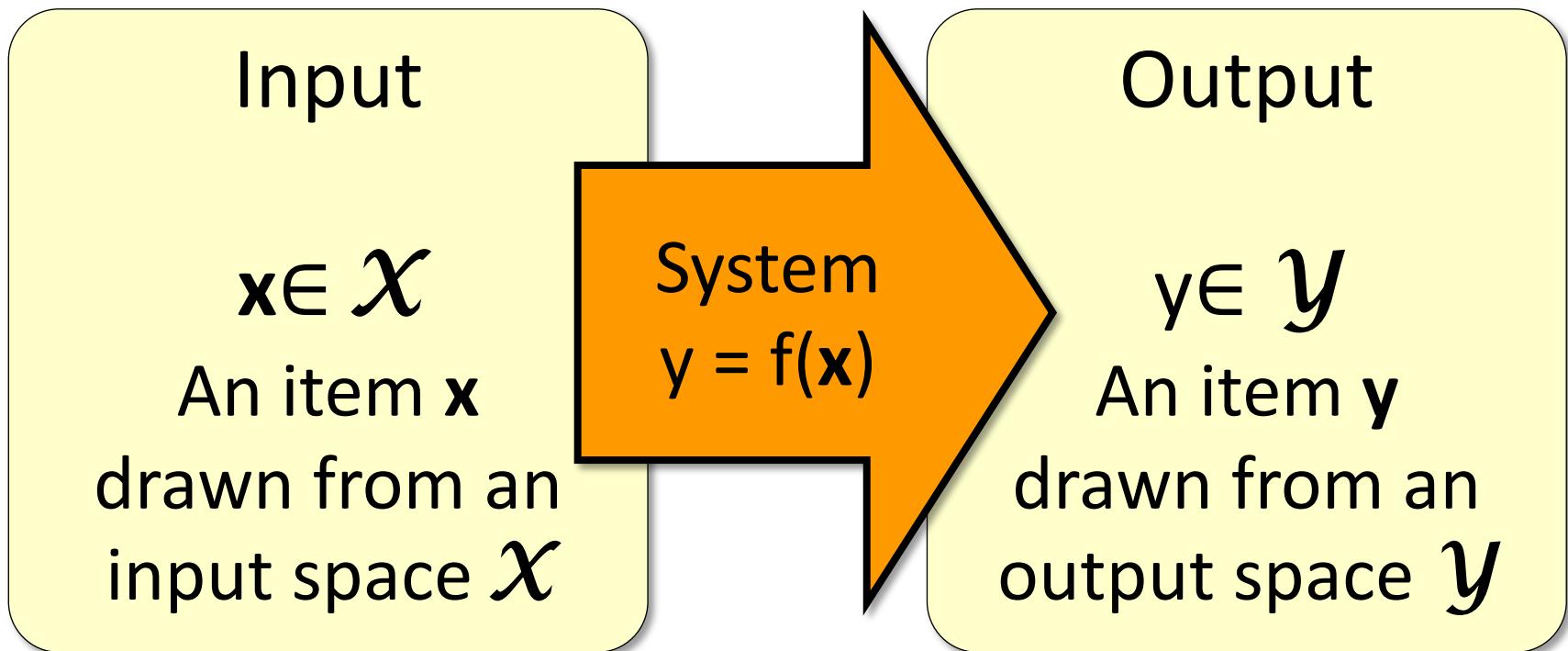
Not lion

Test phase:



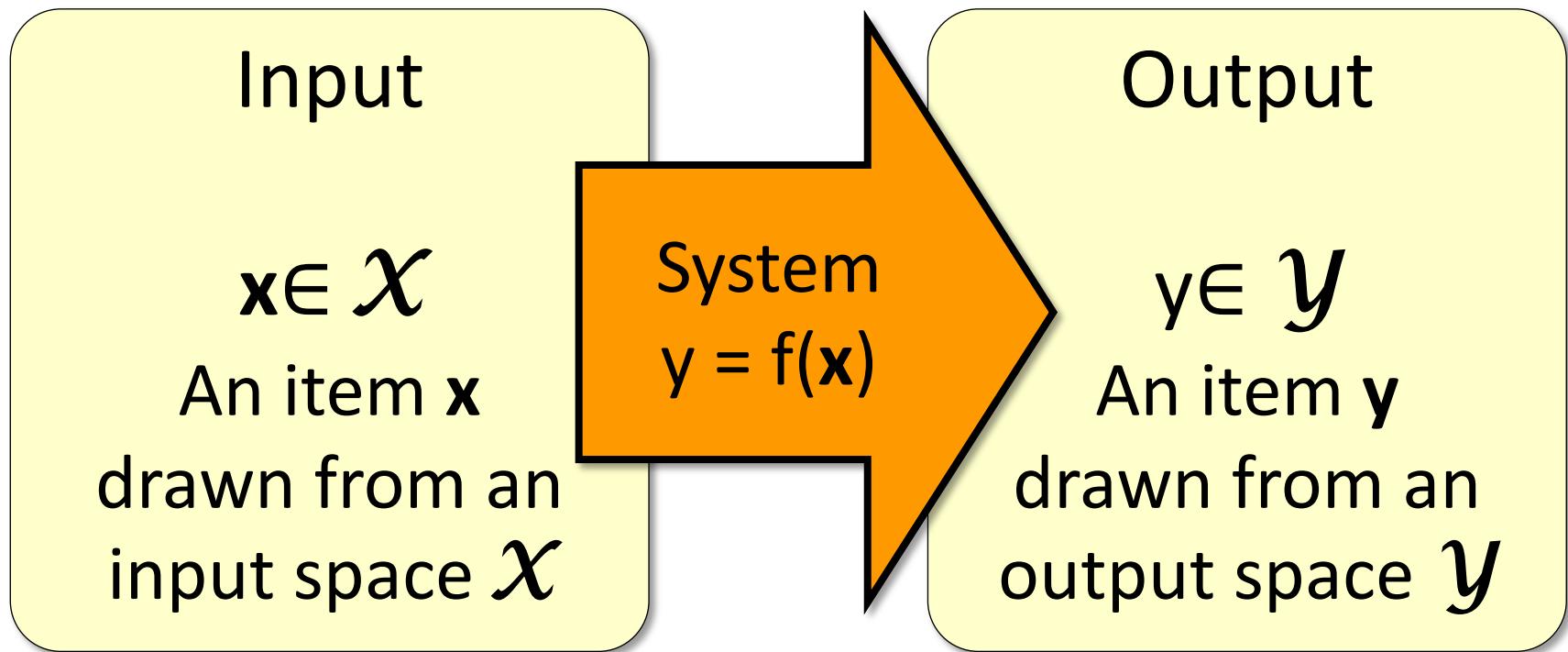
??

Supervised Learning



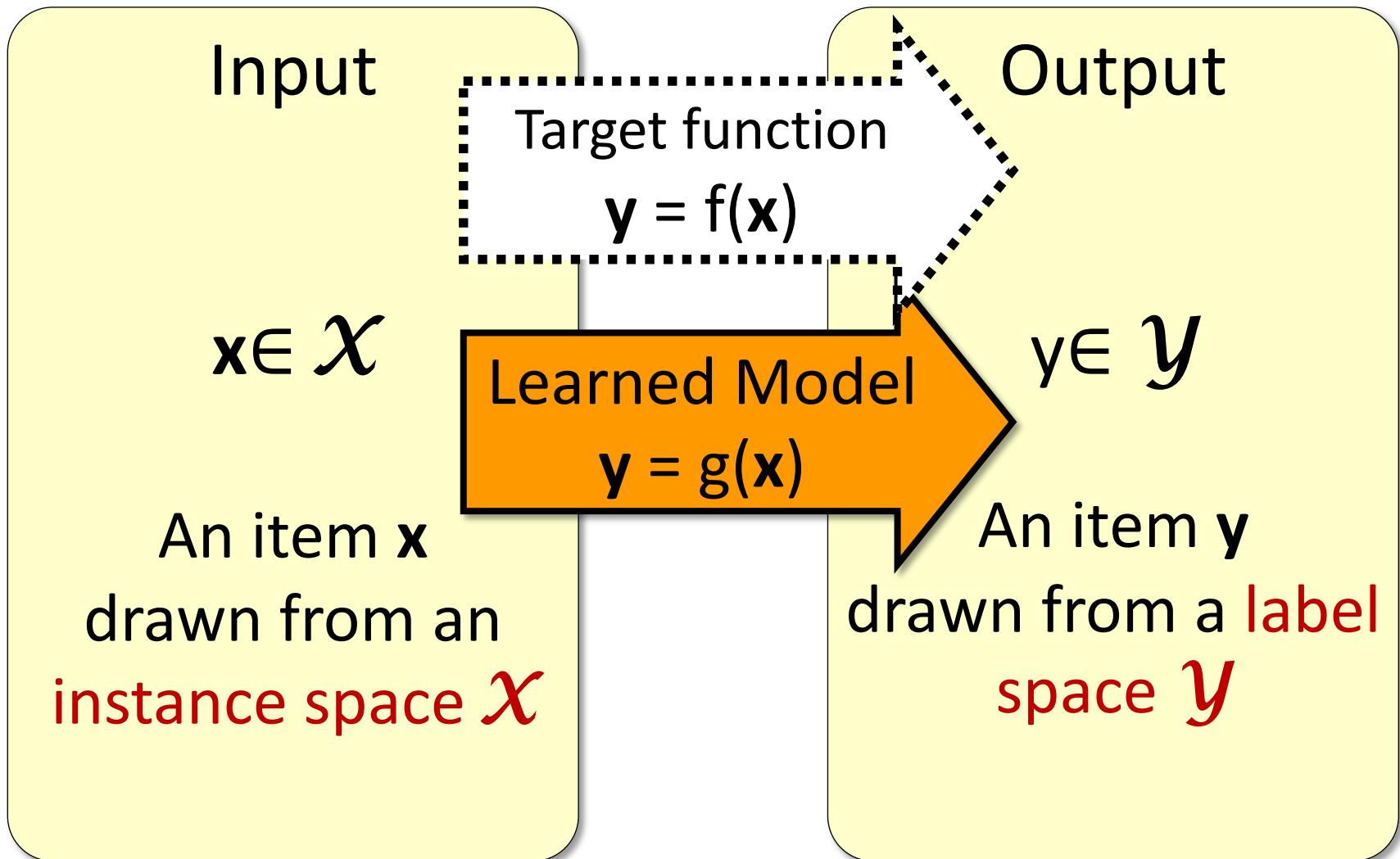
- ❖ We consider systems that apply a function $f()$ to input items x and return an output $y = f(x)$.

Supervised Learning

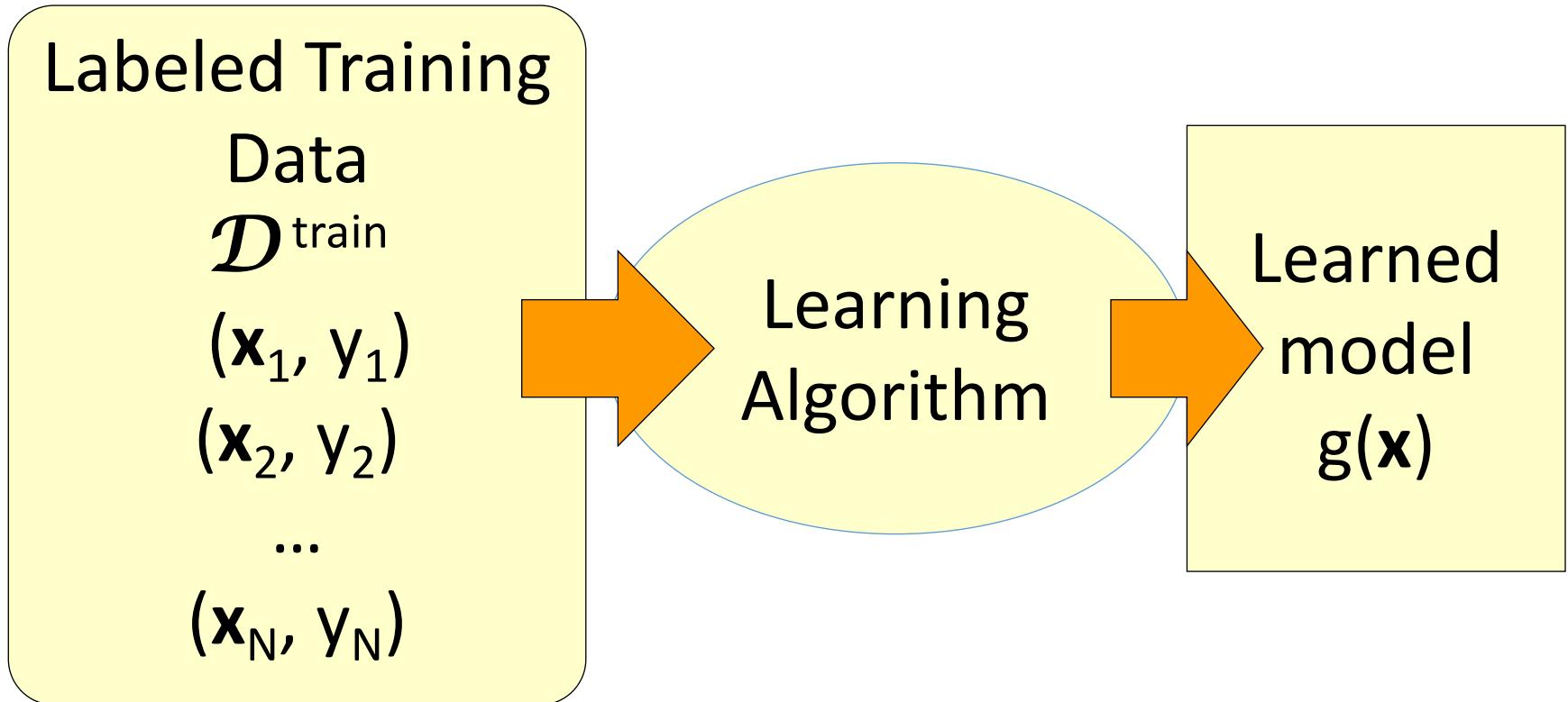


- ❖ In (supervised) machine learning, we deal with systems whose $f(\mathbf{x})$ is learned from examples.

Supervised Learning



Supervised Learning: Training



- ❖ Give the learner examples in $\mathcal{D}^{\text{train}}$
- ❖ The learner returns a model $g(\mathbf{x})$

Supervised Learning: Testing

Labeled
Test Data

$\mathcal{D}^{\text{test}}$

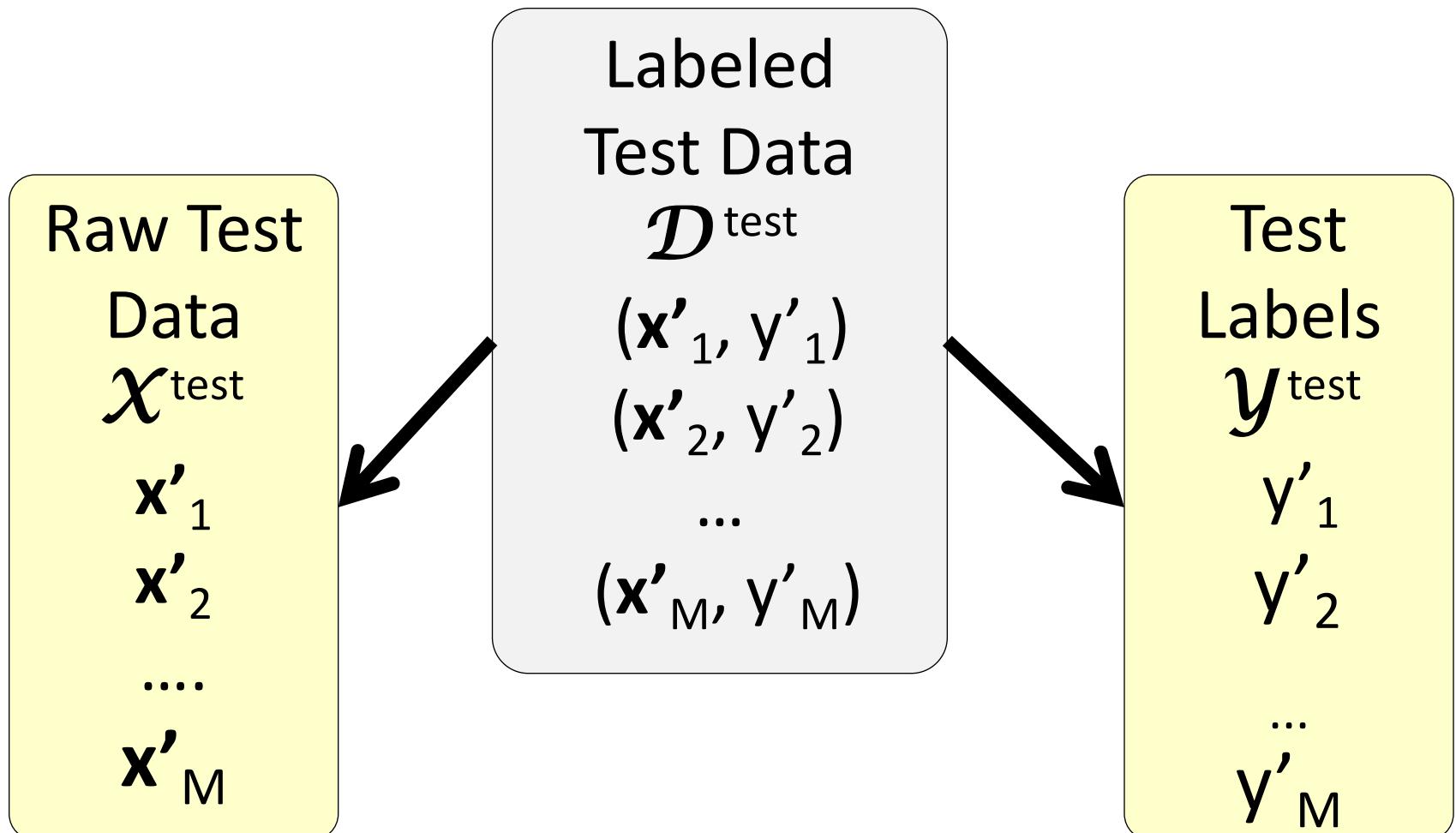
$(\mathbf{x}'_1, \mathbf{y}'_1)$
 $(\mathbf{x}'_2, \mathbf{y}'_2)$

...

$(\mathbf{x}'_M, \mathbf{y}'_M)$

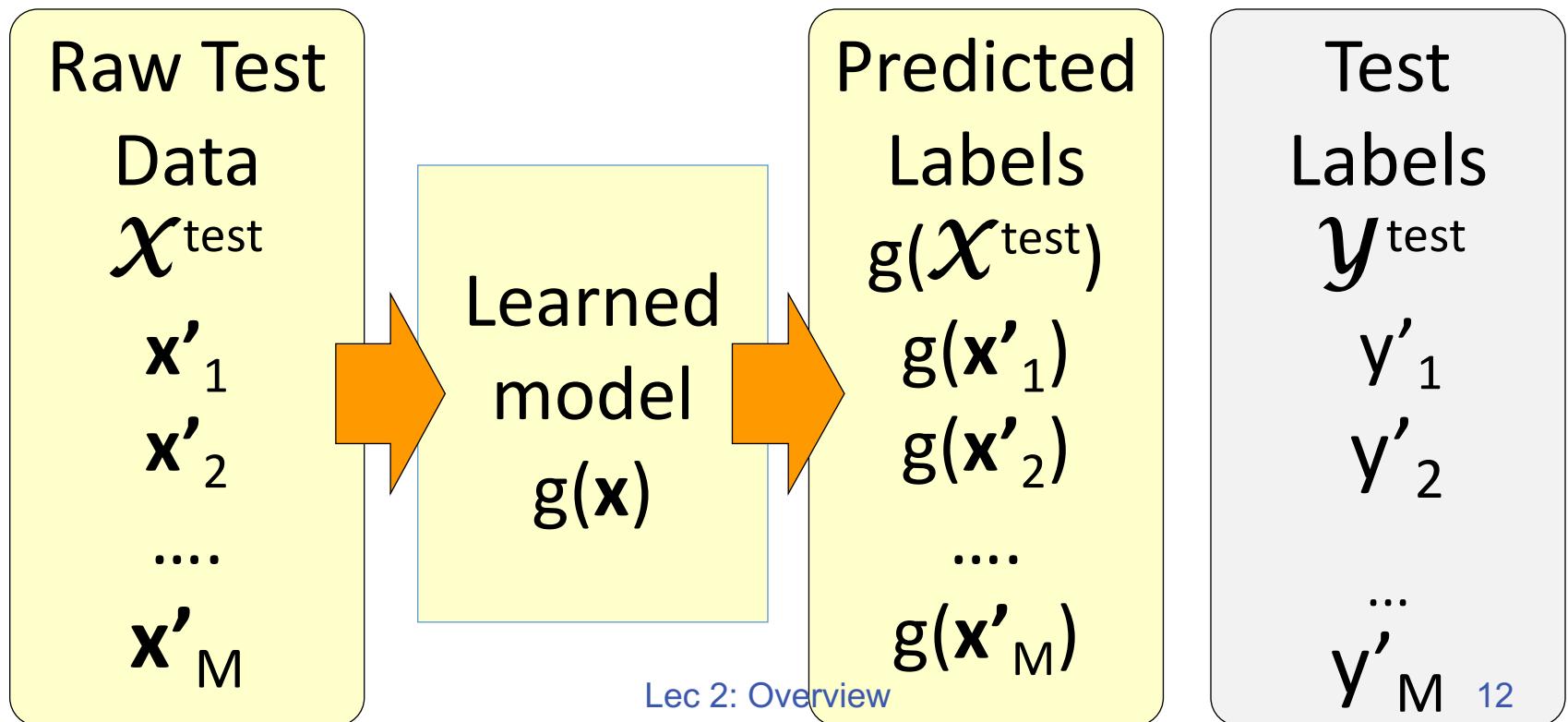
- ❖ Reserve some labeled data for testing

Supervised Learning: Testing



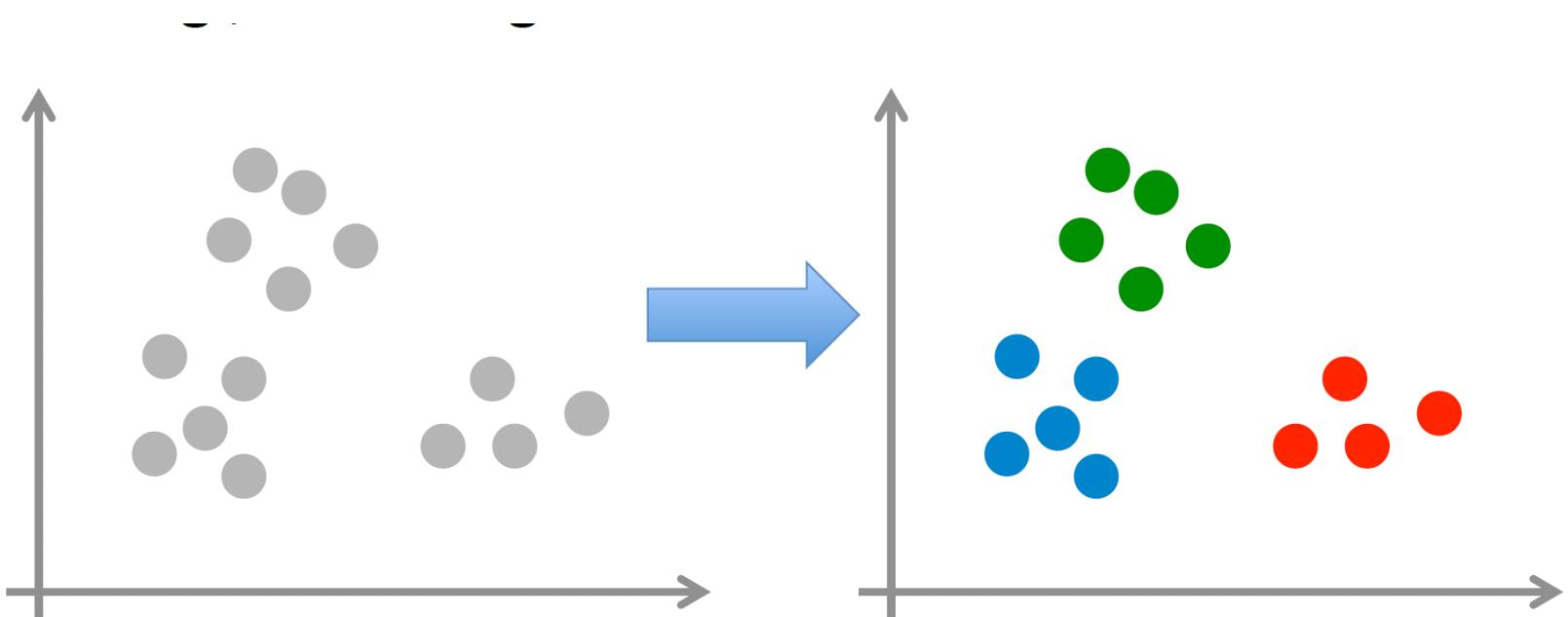
Supervised Learning: Testing

- ❖ Apply the model to the raw test data
- ❖ Evaluate by comparing predicted labels against the test labels

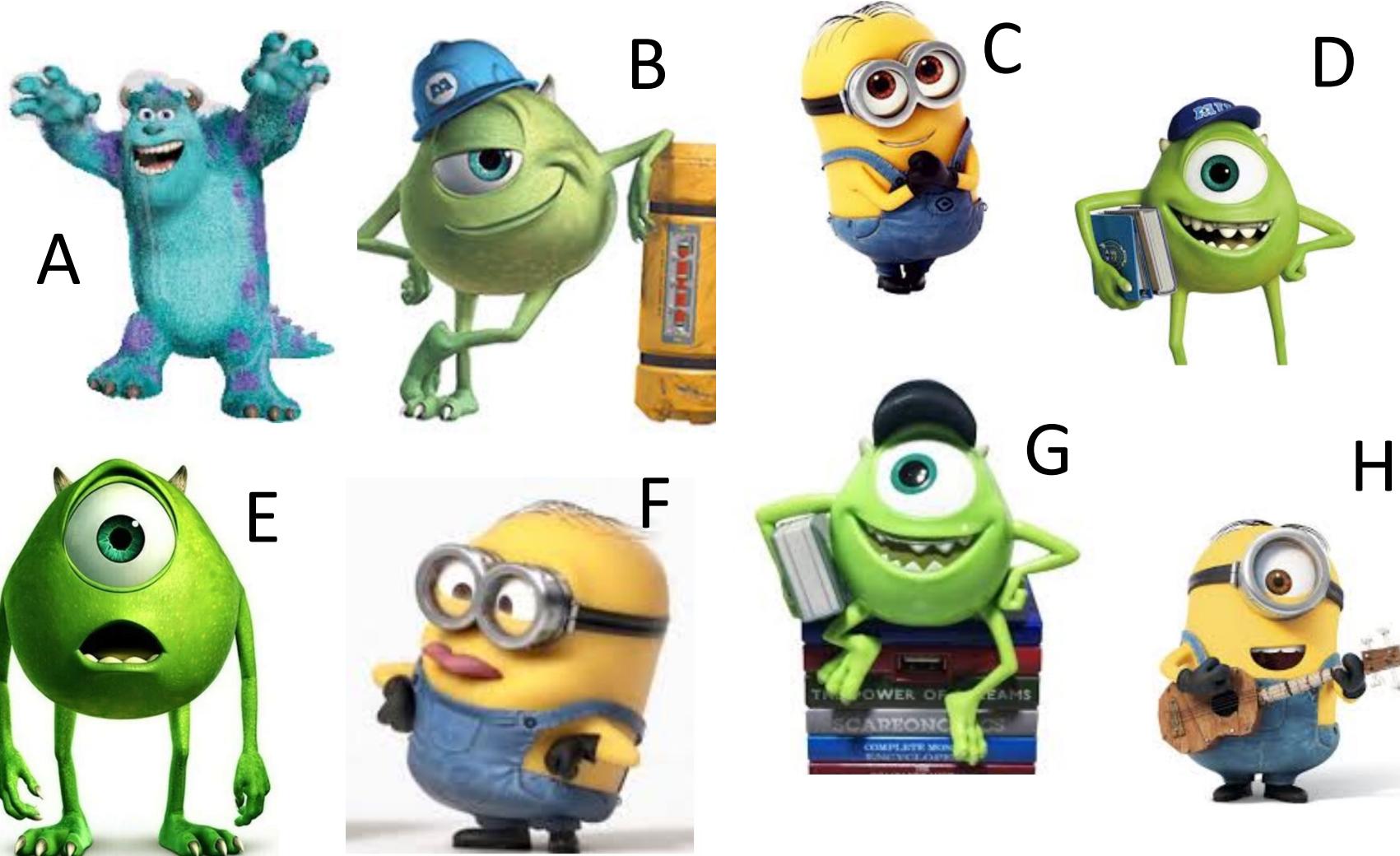


Unsupervised learning

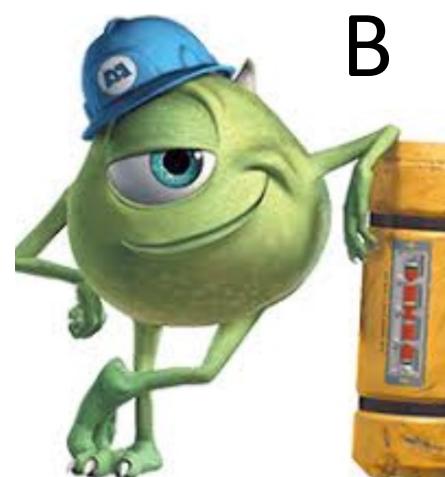
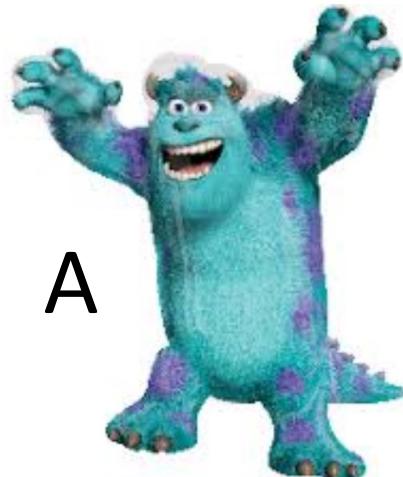
- ❖ Given: **unlabeled** inputs
- ❖ Goal: learn some intrinsic structure in inputs



How many “kinds of monsters” are there?



How many “kinds of monsters” are there?



How many “kinds of monsters” are there?



H



B



C



F



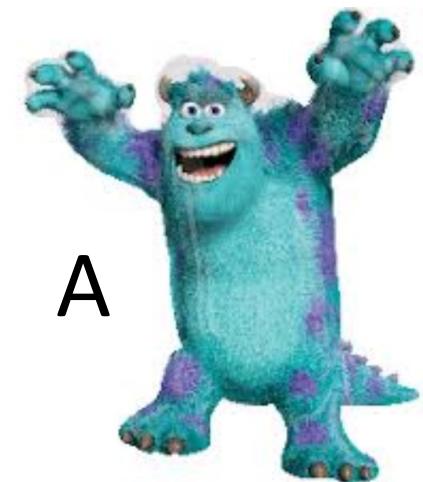
E



G

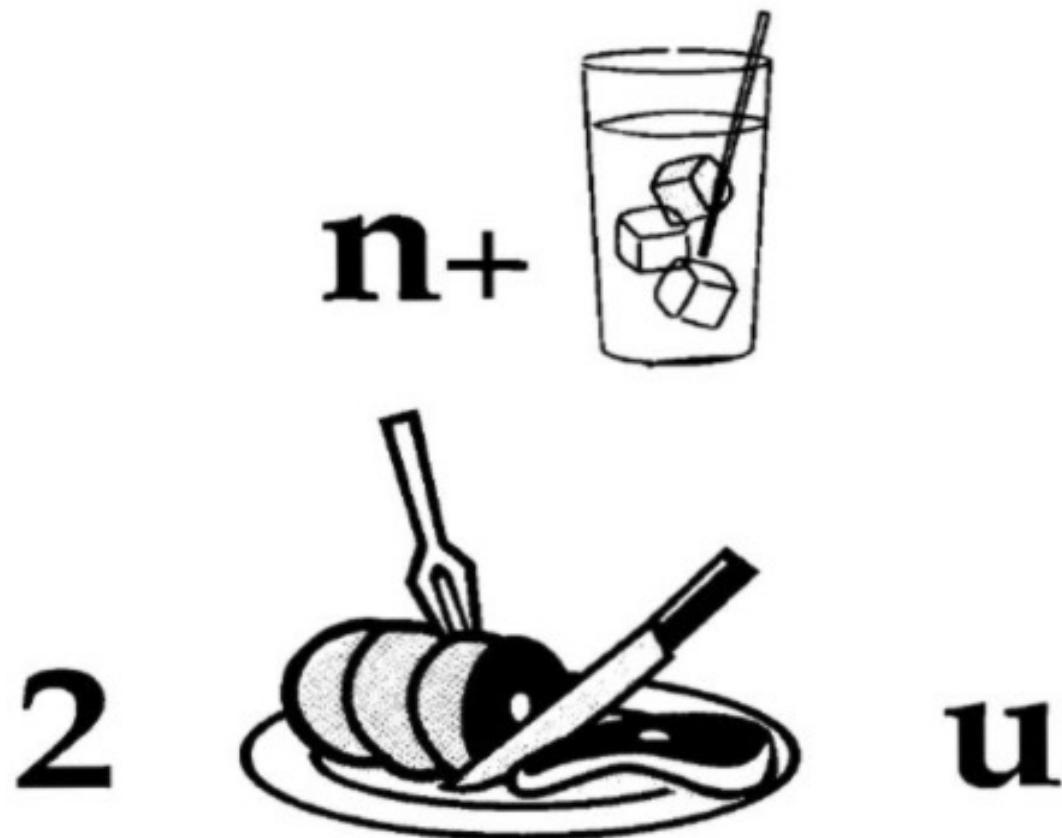


D



A

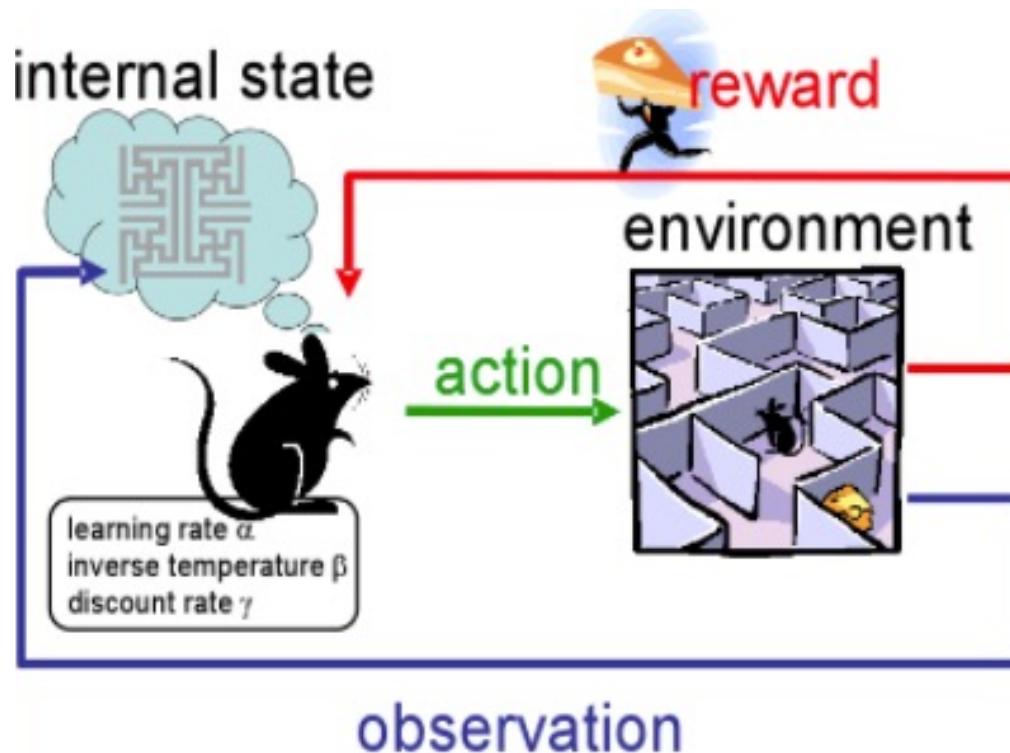
Decipher



Credit: Dan Roth

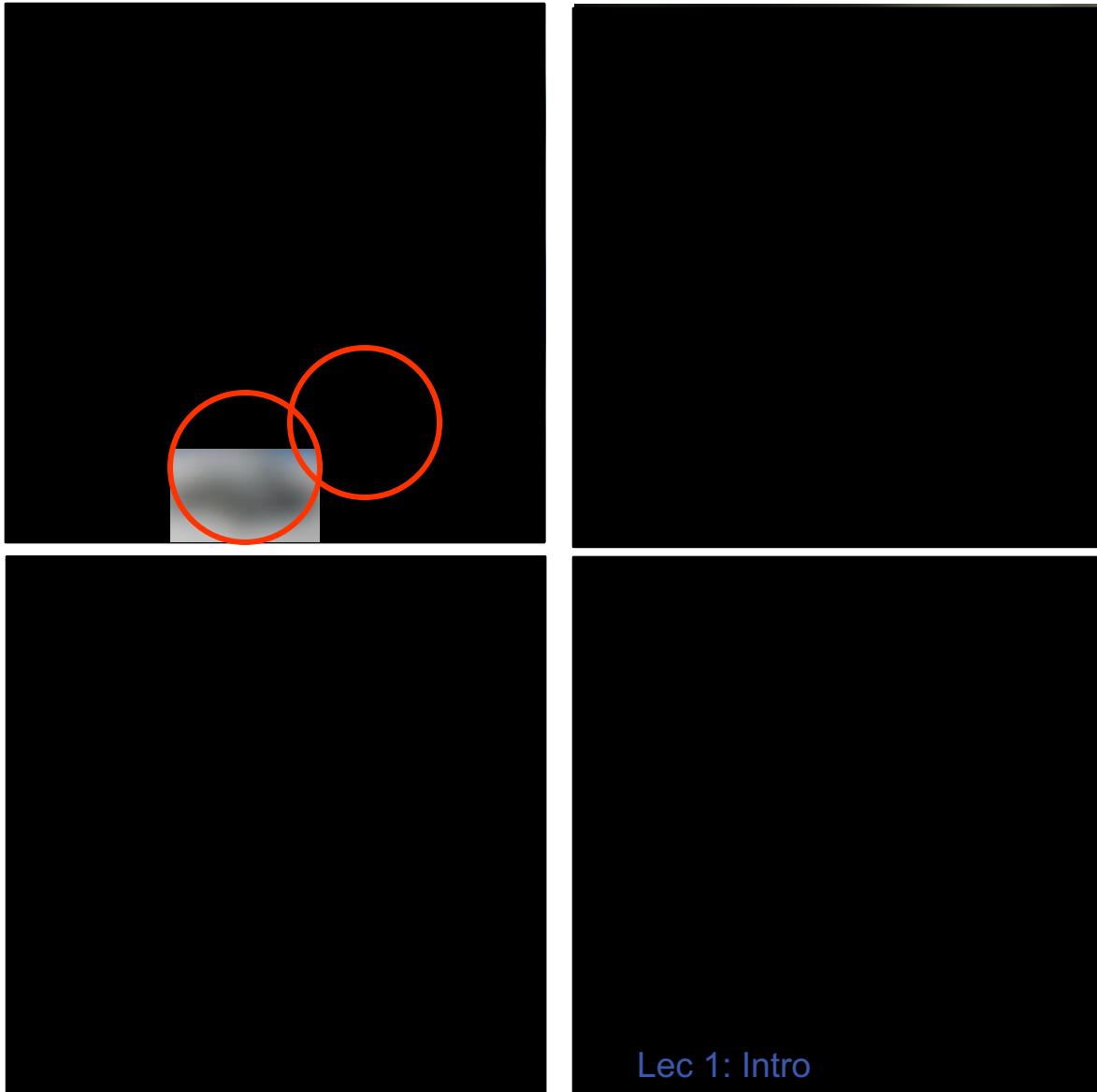
Reinforcement Learning

- ❖ Given sequences of states and actions with rewards
- ❖ Learn policy that maximizes agent's reward



Challenges in ML

Structured Inference



[Credit: Dhruv Batra](#)

Robustness

Car or shoe?



Adversarial Attack



93%, 20 Km/h Sign

+ ϵx



$sign(\nabla * J(\theta, x, y))$

=



90%, 80 Km/h Sign



<https://arxiv.org/abs/1712.09327v1>

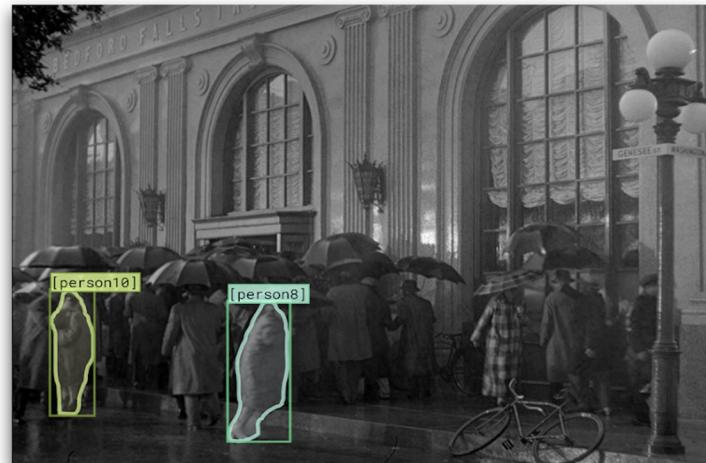
Commonsense

❖ Winograd Schema (1972)

The city councilmen refused the demonstrators a permit because they feared violence.

The city councilmen refused the demonstrators a permit because they advocated violence.

❖ Visual Commonsense



Is it raining outside?

- a) Yes, it is snowing.
- b) Yes, [person8] and [person10] are outside.
- c) No, it looks to be fall.
- d) Yes, it is raining heavily.

Fairness/Inclusion in ML

Select photo ?

X The photo you want to upload does not meet our criteria because:
• Subject eyes are closed

Please refer to the technical requirements.
You have 9 attempts left.

Check the photo [requirements](#).

Read more about [common photo problems](#) and [how to resolve them](#).

After your tenth attempt you will need to

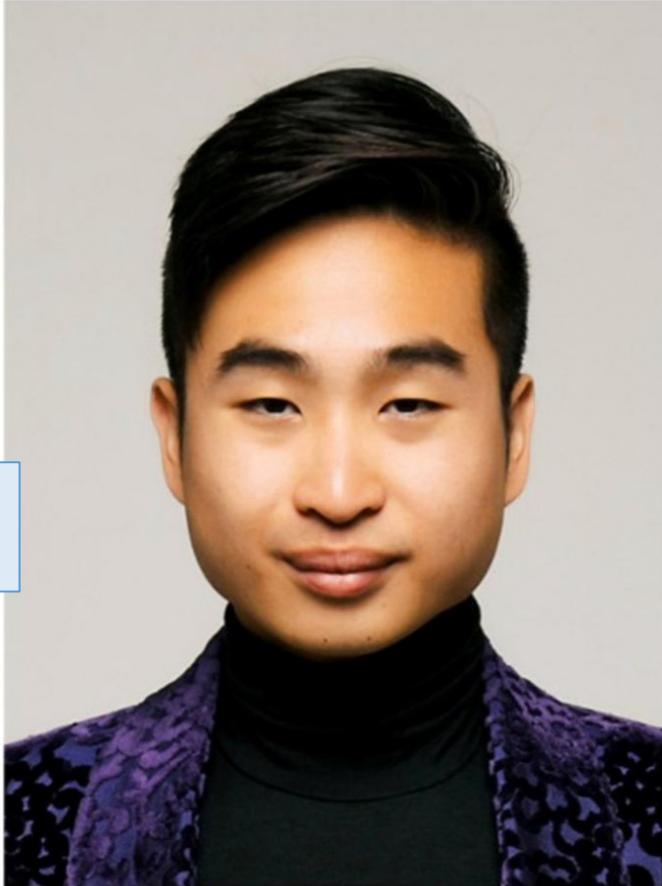
Subject eyes are closed

Reference number: 20161206-81

Filename: Untitled.jpg

If you wish to [contact us](#) about the photo, you must provide us with the reference number given above.

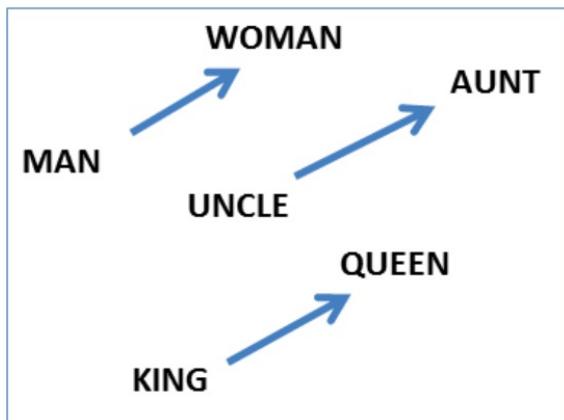
Please print this information for your records.



A screenshot of New Zealand man Richard Lee's passport photo rejection notice, supplied to Reuters December 7, 2016. Richard Lee/Handout via REUTERS

Fairness in ML-- Word embedding bias

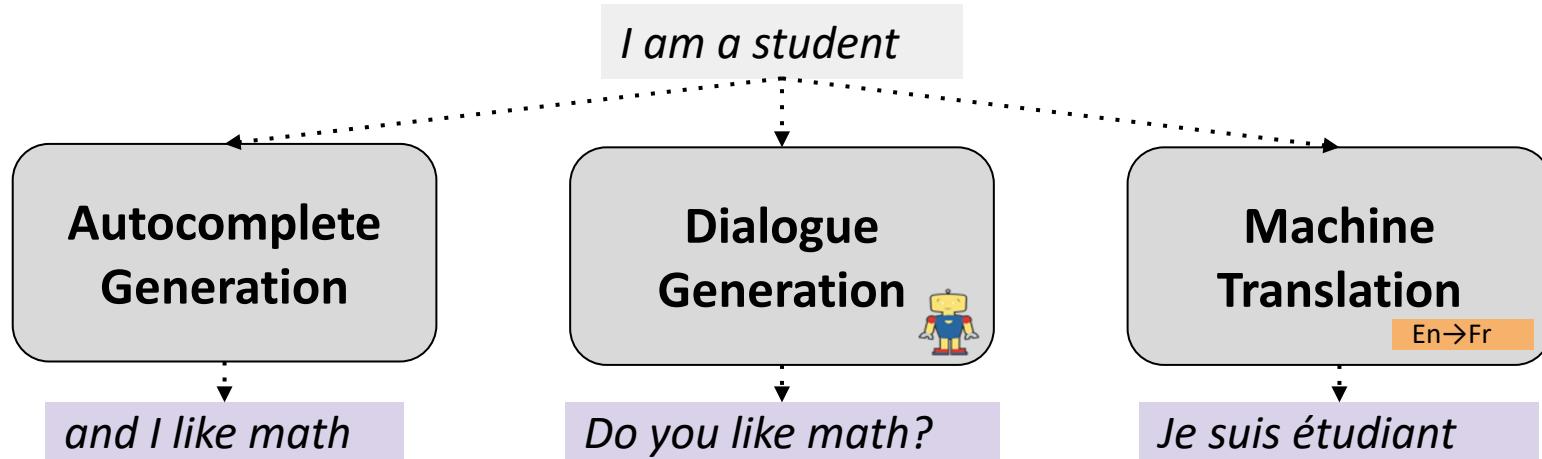
❖ $v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$



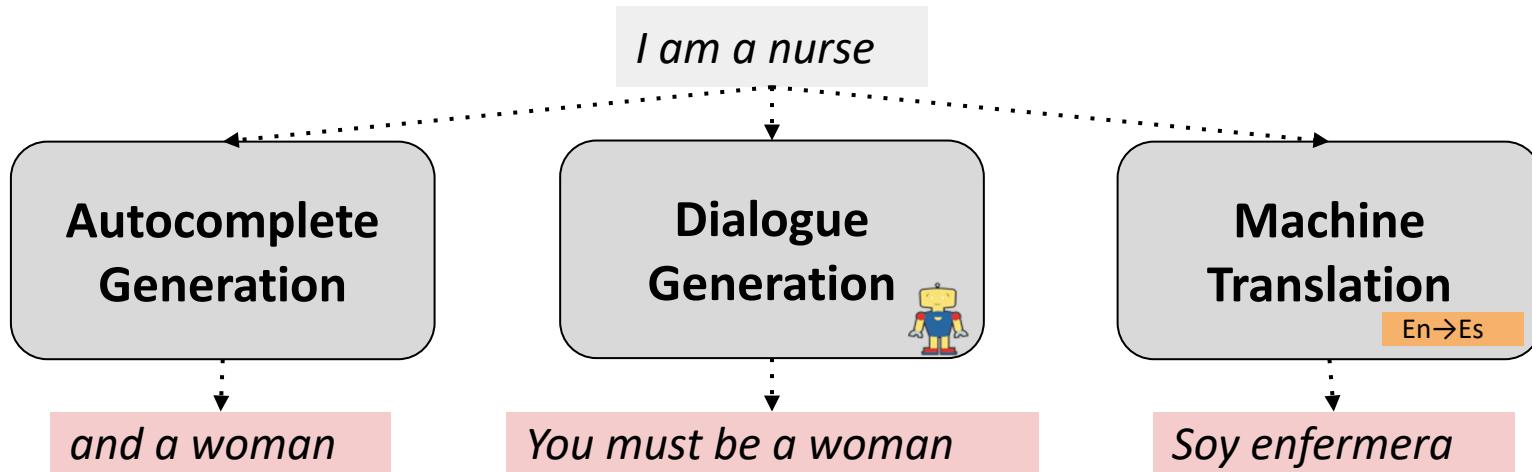
he: __	she: __
uncle	aunt
lion	
surgeon	
architect	
beer	
professor	



We use Google w2v embedding trained from the news



Language generations can be gendered!



Societal Biases in Language Generation: Progress and Challenges

Misgendering in NLG

Alex went to the hospital for their appointment. [MASK] felt sick.

Prediction	Score
Alex went to the hospital for their appointment . She felt sick .	45.3%
Alex went to the hospital for their appointment . He felt sick .	36%
Alex went to the hospital for their appointment . Alex felt sick .	5.8%
Alex went to the hospital for their appointment . I felt sick .	2.3%
Alex went to the hospital for their appointment . They felt sick .	0.4%

<https://demo.allennlp.org/masked-lm>

Framing a Learning Problem

How we set up a learning problem

- ❖ The Badges Game.....
 - ❖ This is an example of the key learning protocol:
supervised learning

The Badges game

+ Naoki Abe

- Eric Baum

- ❖ Conference attendees to the 1994 Machine Learning conference were given **name badges labeled with + or -**.
- ❖ What function was used to assign these labels?

Training data

- | | | |
|---------------------|-------------------|--------------------|
| + Naoki Abe | + Peter Bartlett | + Carla E. Brodley |
| - Myriam Abramson | - Eric Baum | + Nader Bshouty |
| + David W. Aha | + Welton Becket | - Wray Buntine |
| + Kamal M. Ali | - Shai Ben-David | - Andrey Burago |
| - Eric Allender | + George Berg | + Tom Bylander |
| + Dana Angluin | + Neil Berkman | + Bill Byrne |
| - Chidanand Apte | + Malini Bhandaru | - Claire Cardie |
| + Minoru Asada | + Bir Bhanu | + John Case |
| + Lars Asker | + Reinhard Blasig | + Jason Catlett |
| + Javed Aslam | - Avrim Blum | - Philip Chan |
| + Jose L. Balcazar | - Anselm Blumer | - Zhixiang Chen |
| - Cristina Baroglio | + Justin Boyan | - Chris Darken |

Raw test data

Gerald F. DeJong

Chris Drummond

Yolanda Gil

Attilio Giordana

Jiarong Hong

J. R. Quinlan

Priscilla Rasmussen

Dan Roth

Yoram Singer

Lyle H. Ungar

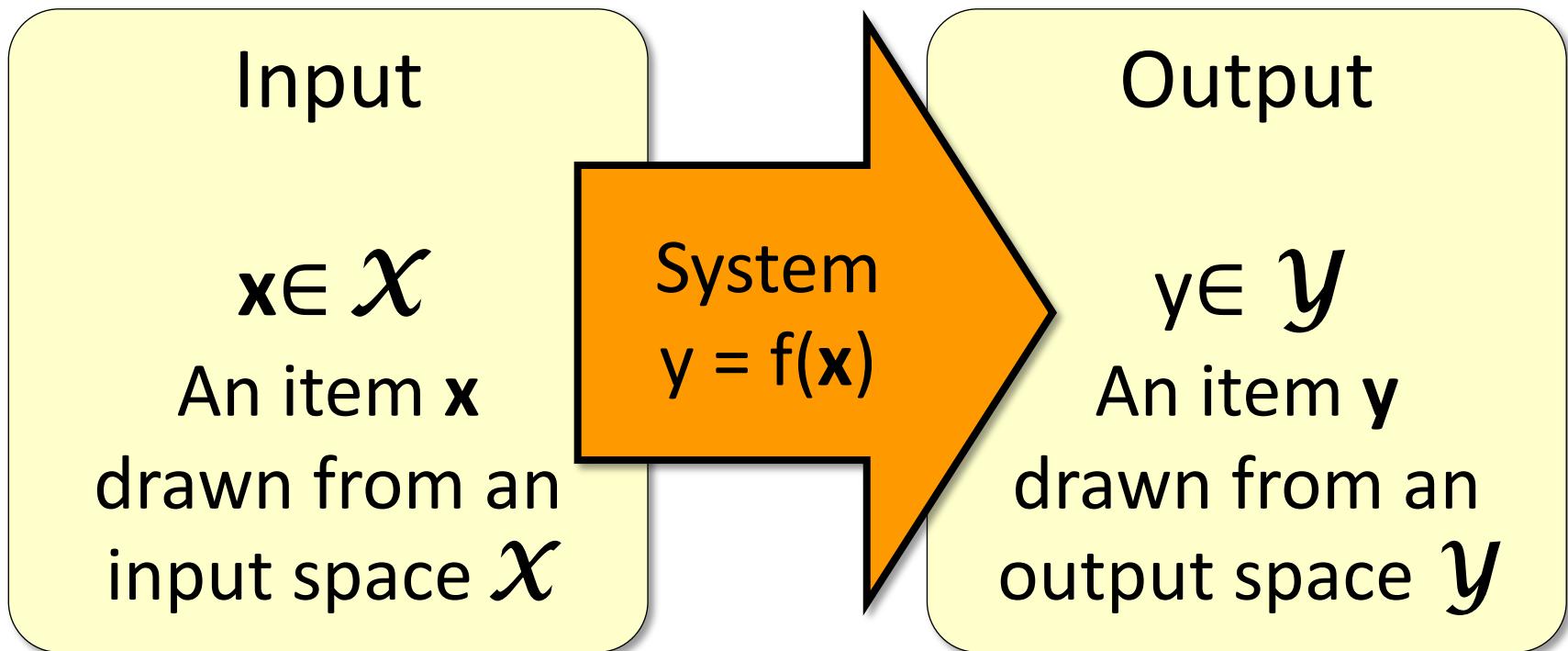
Labeled test data

- + Gerald F. DeJong
- Chris Drummond
- + Yolanda Gil
- Attilio Giordana
- + Jiarong Hong
- J. R. Quinlan
- Priscilla Rasmussen
- + Dan Roth
- + Yoram Singer
- Lyle H. Ungar

Exercise: What is the rule?

- | | | |
|---------------------|-------------------|--------------------|
| + Naoki Abe | + Peter Bartlett | + Carla E. Brodley |
| - Myriam Abramson | - Eric Baum | + Nader Bshouty |
| + David W. Aha | + Welton Becket | - Wray Buntine |
| + Kamal M. Ali | - Shai Ben-David | - Andrey Burago |
| - Eric Allender | + George Berg | + Tom Bylander |
| + Dana Angluin | + Neil Berkman | + Bill Byrne |
| - Chidanand Apte | + Malini Bhandaru | - Claire Cardie |
| + Minoru Asada | + Bir Bhanu | + John Case |
| + Lars Asker | + Reinhard Blasig | + Jason Catlett |
| + Javed Aslam | - Avrim Blum | - Philip Chan |
| + Jose L. Balcazar | - Anselm Blumer | - Zhixiang Chen |
| - Cristina Baroglio | + Justin Boyan | - Chris Darken |

Supervised Learning



- ❖ We consider systems that apply a function $f()$ to input items x and return an output $y = f(x)$.

Using supervised learning

- ❖ What is our **instance space**?
 - ❖ Gloss: What kind of features are we using?
- ❖ What is our **label space**?
 - ❖ Gloss: What kind of learning task are we dealing with?
- ❖ What is our **hypothesis space**?
 - ❖ Gloss: What kind of functions (models) are we learning?
- ❖ What **learning algorithm** do we use?
 - ❖ Gloss: How do we learn the model from the labeled data?
- ❖ What is our **loss function**/evaluation metric?
 - ❖ Gloss: How do we measure success? What drives learning?

1. Input: The instance space \mathcal{X}

Input

$x \in \mathcal{X}$

An item x
drawn from an
instance space
 \mathcal{X}

x is represented in a **feature space**

- Typically $x \in \{0,1\}^n$ or R^N
- Usually represented as a **vector**
- We call it **input vector**

Example:

Boolean features:

Does this email contain the word ‘money’?

Numerical features:

How often does ‘money’ occur in this email

What is the width/height of this bounding box?

What is the length of the first name?

What's χ for the Badges game?

❖ Possible features:

- Length of their first or last name?
- Does the name contain letter 'x'?
- How many vowels does their name contain?
- Is the n-th letter a vowel?

+ Naoki Abe

- Myriam Abramson

+ David W. Aha

+ Kamal M. Ali

- Eric Allender

+ Dana Angluin

+ Peter Bartlett

- Eric Baum

+ Welton Becket

- Shai Ben-David

+ George Berg

+ Neil Berkman

+ Carla E. Brodley

+ Nader Bshouty

- Wray Buntine

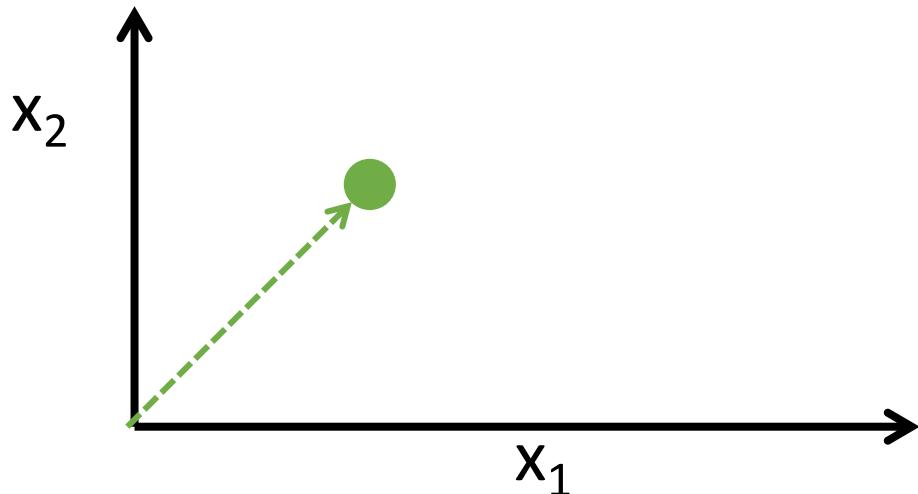
- Andrey Burago

+ Tom Bylander

+ Bill Byrne

\mathcal{X} as a vector space

- ❖ \mathcal{X} is an N-dimensional vector space (e.g. \mathbb{R}^N)
 - ❖ Each dimension = one feature.
- ❖ Each \mathbf{x} is a **feature vector** (hence the boldface \mathbf{x}).
- ❖ Think of $\mathbf{x} = [x_1 \dots x_N]$ as a point in \mathcal{X} :



Example: the badge game

+ Naoki Abe

- Myriam Abramson

+ David W. Aha

+ Kamal M. Ali

- Eric Allender

+ Dana Angluin

+ Peter Bartlett

- Eric Baum

+ Welton Becket

- Shai Ben-David

+ George Berg

+ Neil Berkman

+ Carla E. Brodley

+ Nader Bshouty

- Wray Buntine

- Andrey Burago

+ Tom Bylander

+ Bill Byrne

[first-char is vowel, first-char is A, first-char is N, second-char is vowel ...]

+ Naoki Abe

[0 , 0 , 1 , 1 ...]

- Avrim Blum

[1 , 1 , 0 , 0 ...]

Good features are essential

- ❖ The choice of features is crucial for how well a task can be learned.
 - ❖ In many application areas (language, vision, etc.), a lot of work goes into designing suitable features.
 - ❖ This requires domain expertise.
- ❖ CM146 can't teach you what specific features to use for your task.
 - ❖ But we will touch on some general principles

2. Output space

y is represented in output space
(label space)

Different kinds of output:

- Binary classification:
 $y \in \{-1, 1\}$
- Multiclass classification:
 $y \in \{1, 2, 3, \dots, K\}$
- Regression:
 $y \in R$
- Structured output
 $y \in \{1, 2, 3, \dots, K\}^N$

Output

$$y \in \mathcal{Y}$$

An item y
drawn from a label
space \mathcal{Y}

Supervised Learning : Examples

Animal recognition

- ❖ x : Bitmap picture of the animal
- ❖ y :



Lion? Yes/[No](#)

Lion/Cat/[Dog](#)

Lion/[Mammal](#)/[Dog](#)/Fish

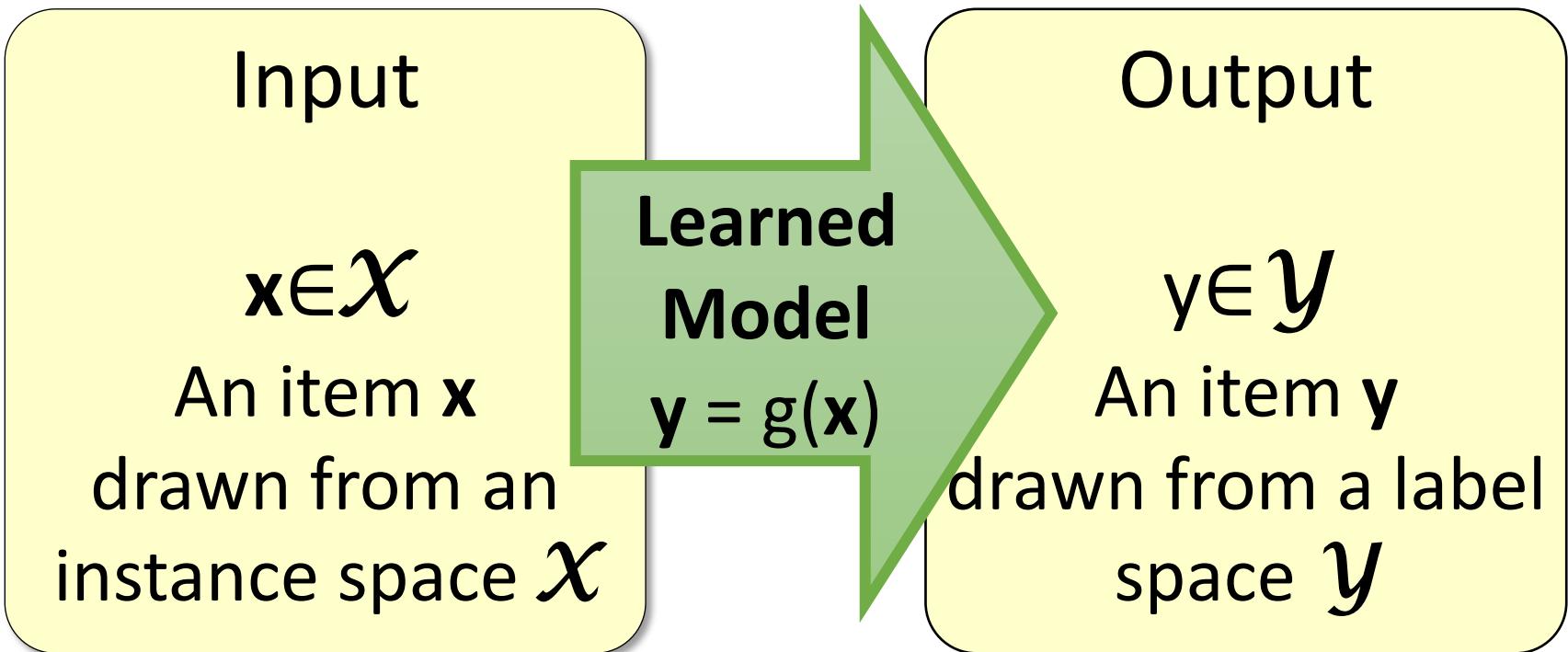
Binary output

Multiclass output

Multilabel output

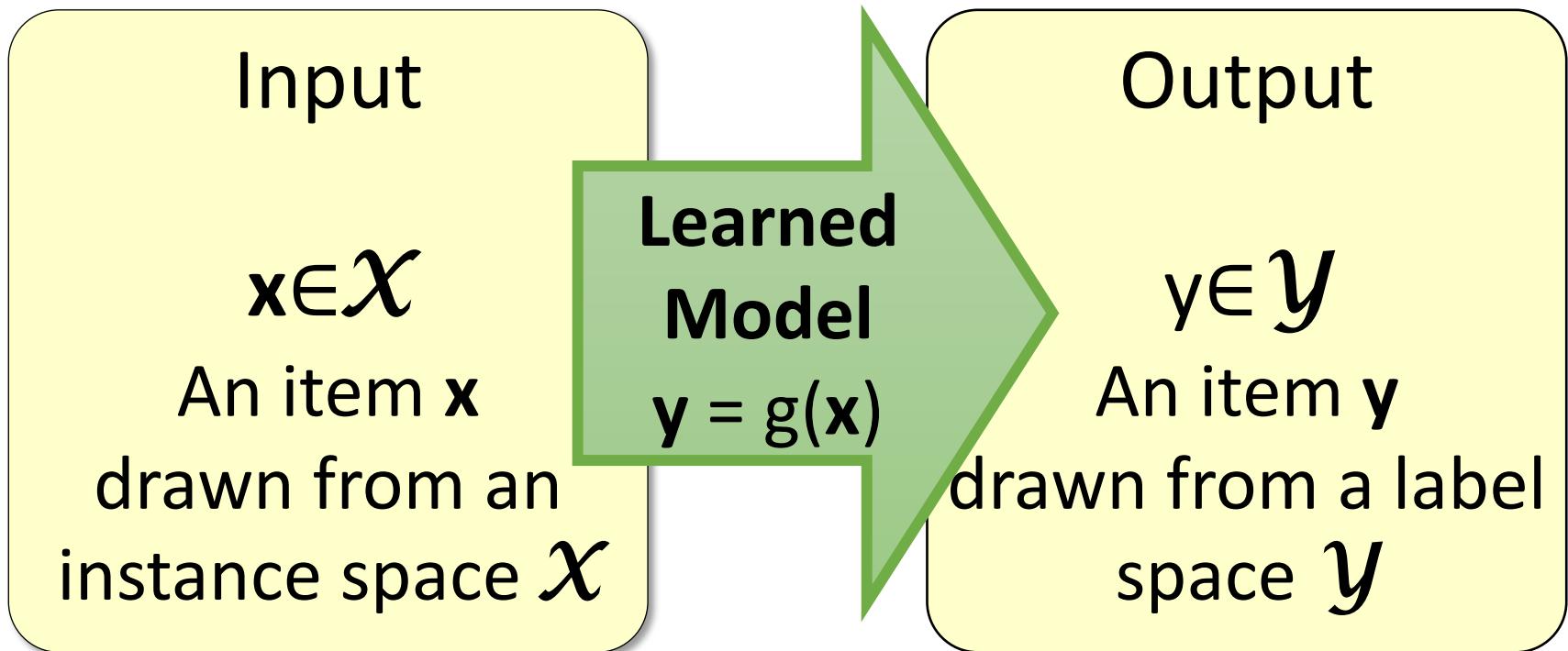
Output of the applications may different from the output of ML models.

3. The model $g(\mathbf{x})$



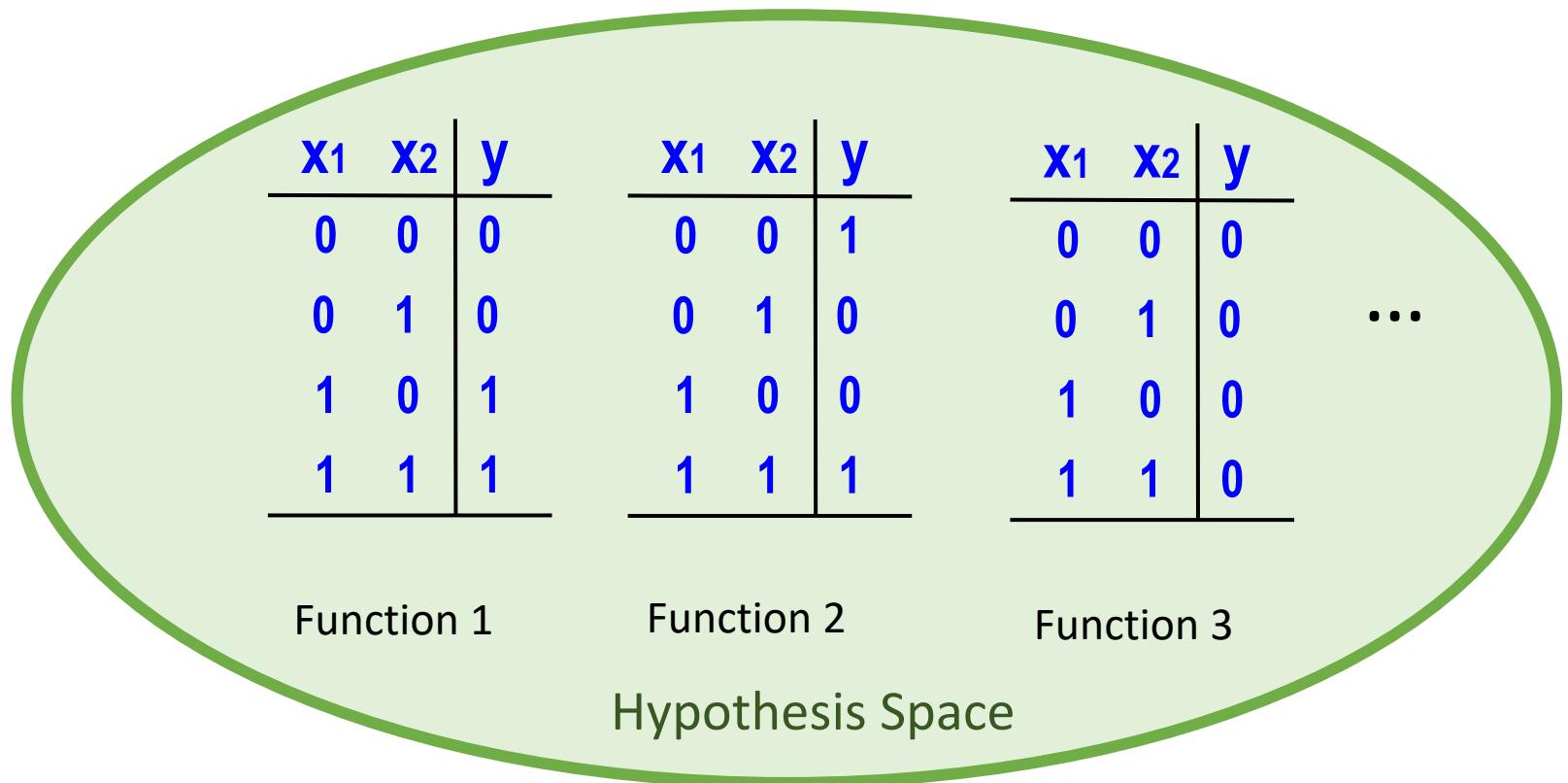
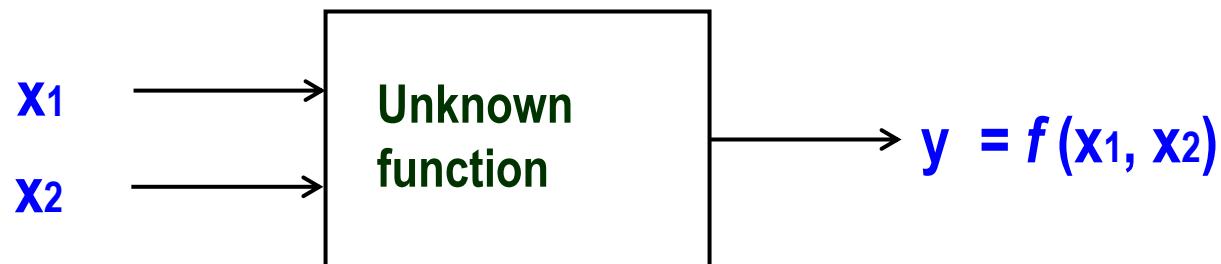
- ❖ We need to choose what *kind* of model we want to learn

3. The model $g(\mathbf{x})$

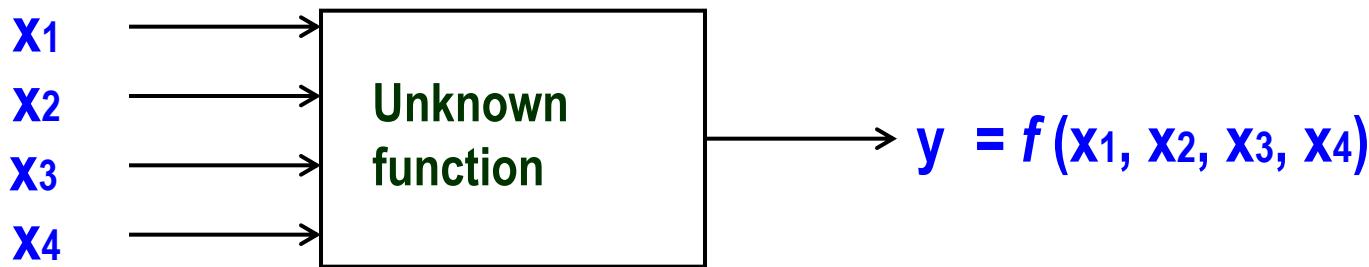


- ❖ We need to choose what *kind* of model we want to learn

Boolean Function



A Learning Problem



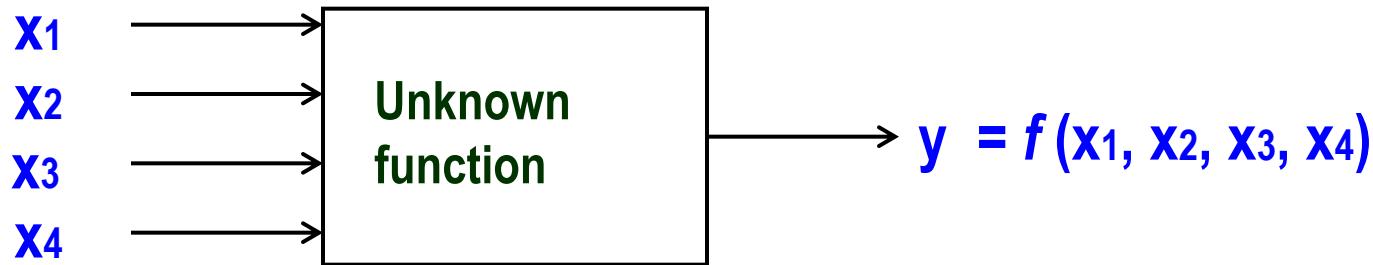
Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

A function g is consistent to a dataset
 $D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$

Discussion: A Learning Problem



Example	x_1	x_2	x_3	x_4	y	
1	0	0	1	0	0	Can you learn this function?
2	0	1	0	0	0	What is it?
3	0	0	1	1	1	A function g is consistent to a dataset
4	1	0	0	1	1	$D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$
5	0	1	1	0	0	
6	1	1	0	0	0	How many possible functions over four features?
7	0	1	0	1	0	How many function is consistent to D on the left

Hypothesis Space

How many possible functions over four features?

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

Example	x1	x2	x3	x4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	0	1	0	?
0	0	0	1	1	?
0	1	0	0	0	?
0	1	0	0	1	?
0	1	1	0	0	?
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	?
1	0	0	1	0	?
1	0	1	1	1	?
1	1	0	0	0	?
1	1	0	0	1	?
1	1	1	1	0	?
1	1	1	1	1	?

Hypothesis Space

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Is Learning Possible?

which one is the most likely one?

Example	X1	X2	X3	X4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	1
1	0	1	0	0	?
1	0	1	1	1	?
1	1	0	0	0	0
1	1	0	1	1	?
1	1	1	0	1	?
1	1	1	1	0	?
1	1	1	1	1	?

Hypothesis Space

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

We can
correctly
possibly

After

have 2^T possibilities for T

- There are $|Y|^{|X|}$ possible functions $f(x)$ from the instance space X to the label space Y .

- Learners typically consider **only a subset** of the functions from X to Y , called the hypothesis space H . $H \subseteq |Y|^{|X|}$

Is Learning Possible?

Example	X1	X2	X3	X4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
					1
					0
					0
					?
					1
					0
					0
					?
					1
					?
					?
					1
					?
					?
					0
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space (2)

Simple Rules: **conjunctive rules**

of the form $y = x_i \wedge x_j \wedge \dots \wedge x_k$

e.g., $y = x_2 \wedge x_3$

$y = x_1 \wedge x_2 \wedge X_4$

How large is the hypothesis space?

Hypothesis Space (2)

1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	0	1	1	1	1
4	1	0	0	0	1	1
5	1	0	1	1	0	0
6	1	1	1	0	0	0
7	0	1	0	1	0	0

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=1$		$x_2 \wedge x_3$	
x_1		$x_2 \wedge x_4$	
x_2		$x_3 \wedge x_4$	
x_3		$x_1 \wedge x_2 \wedge x_3$	
x_4		$x_1 \wedge x_2 \wedge x_4$	
$x_1 \wedge x_2$		$x_1 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_3$		$x_2 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_4$		$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	

Hypothesis Space (2)

1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	0	1	1	1	1
4	1	0	0	0	1	1
5	1	0	1	1	0	0
6	1	1	0	0	0	0
7	0	1	0	1	0	0

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=c$		$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
x_2	0100 0	$x_3 \wedge x_4$	1001 1
x_3	0110 0	$x_1 \wedge x_2 \wedge x_3$	0011 1
x_4	0101 1	$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_2$	1100 0	$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

No simple rule explains the data. The same is true for **simple clauses**.

Hypothesis Space (3)

m-of-n rules: There are 32 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>	<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>
{X ₁ }					{X ₂ , X ₄ }				
{X ₂ }					{X ₃ , X ₄ }				
{X ₃ }					{X ₁ , X ₂ , X ₃ }				
{X ₄ }					{X ₁ , X ₂ , X ₄ }				
{X ₁ , X ₂ }					{X ₁ , X ₃ , X ₄ }				
{X ₁ , X ₃ }					{X ₂ , X ₃ , X ₄ }				
{X ₁ , X ₄ }					{X ₁ , X ₂ , X ₃ , X ₄ }				
{X ₂ , X ₃ }									

Hypothesis Space (3)

m-of-n rules: There are 32 possible rules of the form " $y = 1$ if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

variables	1-of	2-of	3-of	4-of	variables	1-of	2-of	3-of	4-of
{X1}	3	-	-	-	{X2, X4}	2	3	-	-
{X2}	2	-	-	-	{X3, X4}	4	4	-	-
{X3}	1	-	-	-	{X1, X2, X3}	1	3	3	-
{X4}	7	-	-	-	{X1, X2, X4}	2	3	3	-
{X1, X2}	2	3	-	-	{X1, X3, X4}	1	***	3	-
{X1, X3}	1	3	-	-	{X2, X3, X4}	1	5	3	-
{X1, X4}	6	3	-	-	{X1, X2, X3, X4}	1	5	3	3
{X2, X3}	2	3	-	-					

Found a consistent hypothesis.
58 Lec 3: Model & KNN

Views of Learning

- ❖ Learning is the removal of our remaining uncertainty:
- ❖ Learning requires guessing a good hypothesis class
 - ❖ Start with a small class and enlarge it until it contains an hypothesis that fits the data.
- ❖ We could be wrong !
 - ❖ Our guess of the hypothesis space could be wrong
 - ❖ $y=x4 \wedge$ one-of (x_1, x_3) is also consistent

General strategies for Machine Learning

- ❖ Develop flexible hypothesis spaces:
 - ❖ Decision trees, neural networks, nested collections.
- ❖ Develop representation languages for restricted classes of functions:
 - ❖ E.g., Functional representation (n-of-m); Grammars; linear functions; stochastic models

General strategies for Machine Learning

- ❖ Develop flexible hypothesis spaces:
 - ❖ Decision trees, neural networks, nested collections.
- ❖ Develop representation languages for restricted classes of functions:

In either case:

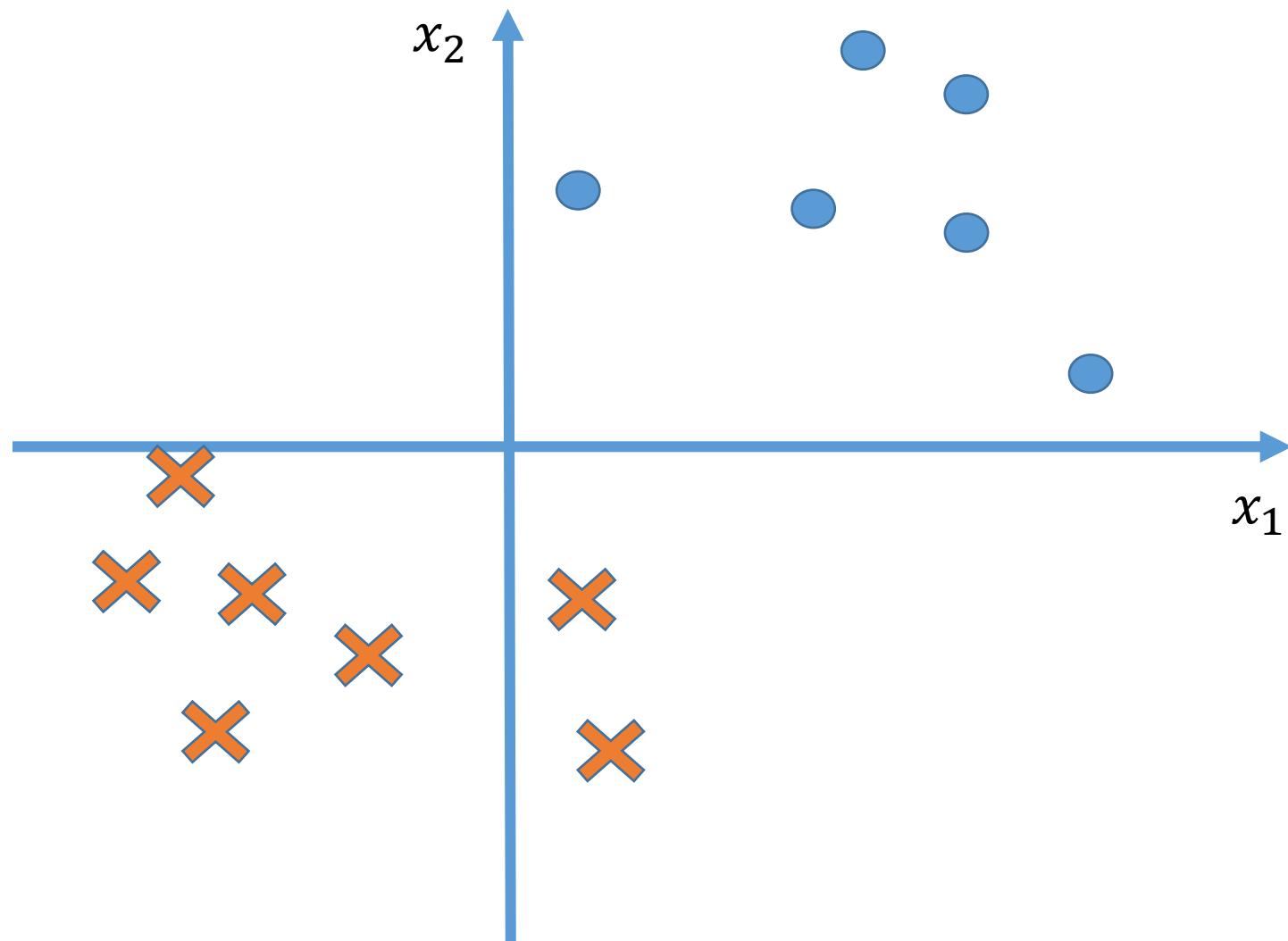
- ❖ Develop algorithms for finding a hypothesis in our hypothesis space, that fits the data
- ❖ And hope that they will generalize well

Hypothesis Space -- Real-Value Features

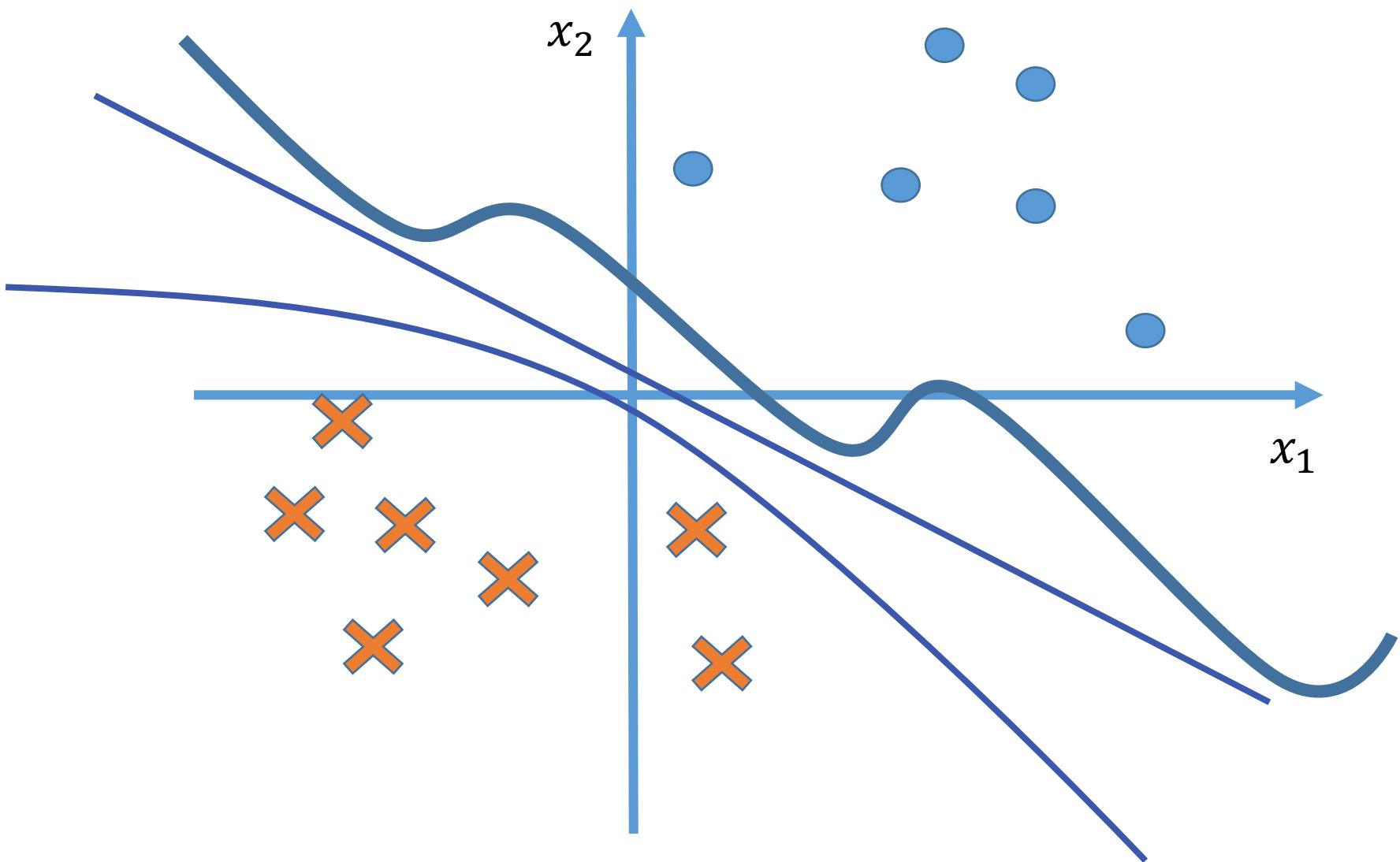
WHAAAAA?!?!



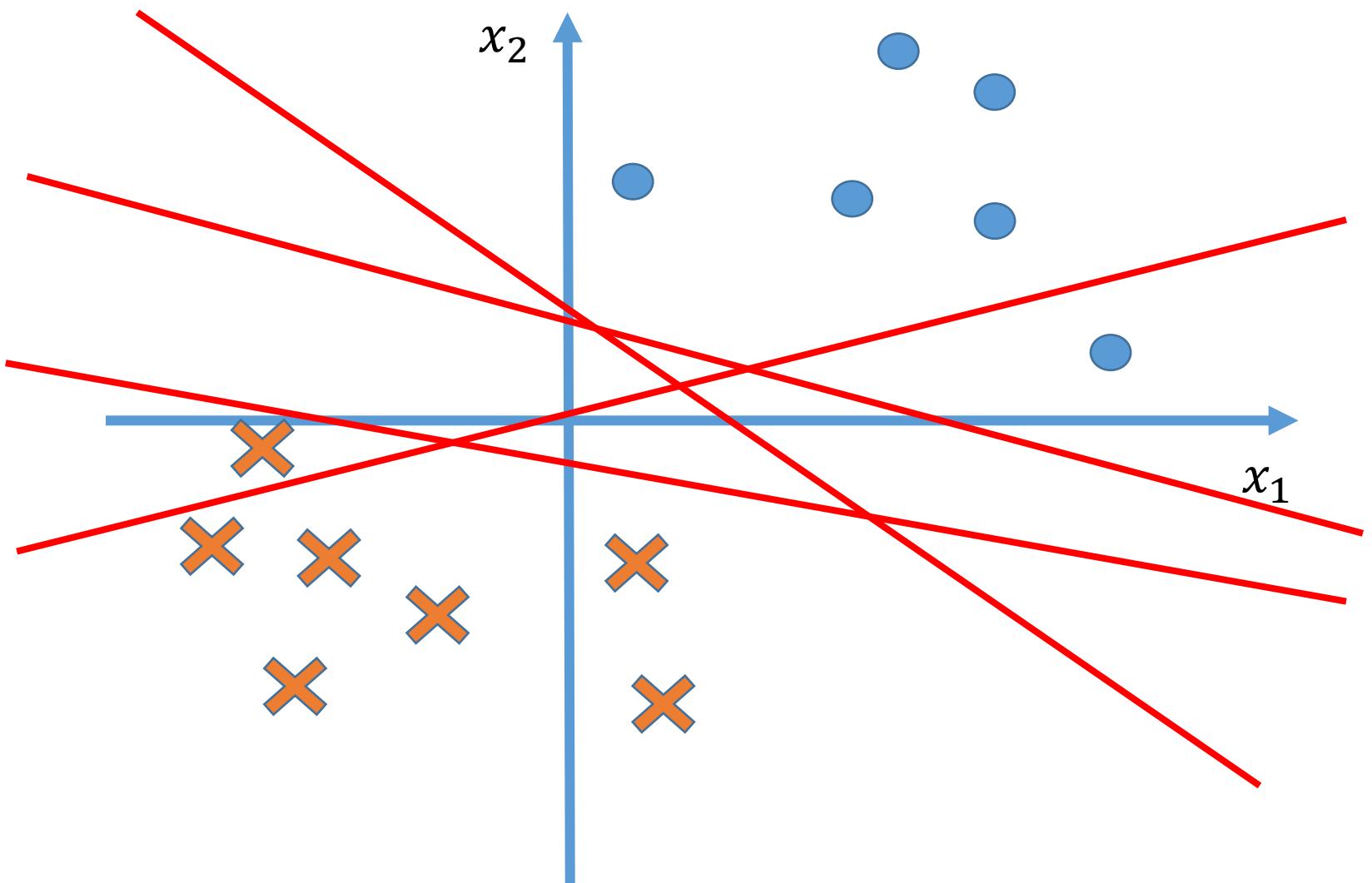
Example problem



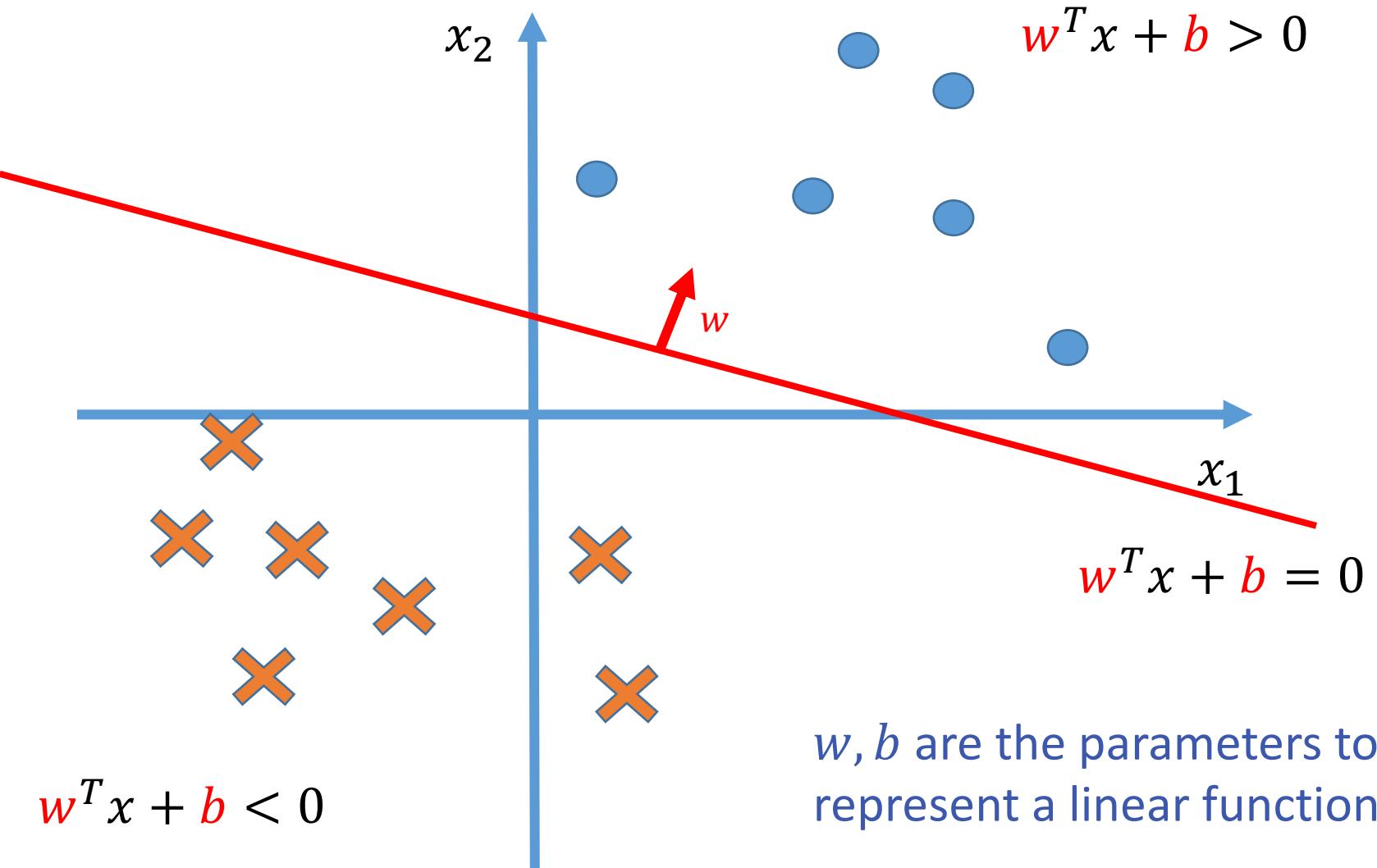
Hypothesis space:



Hypothesis space: linear model



Hypothesis space: linear model



Lecture 3:

Hypothesis Space & KNN

Fall 2022

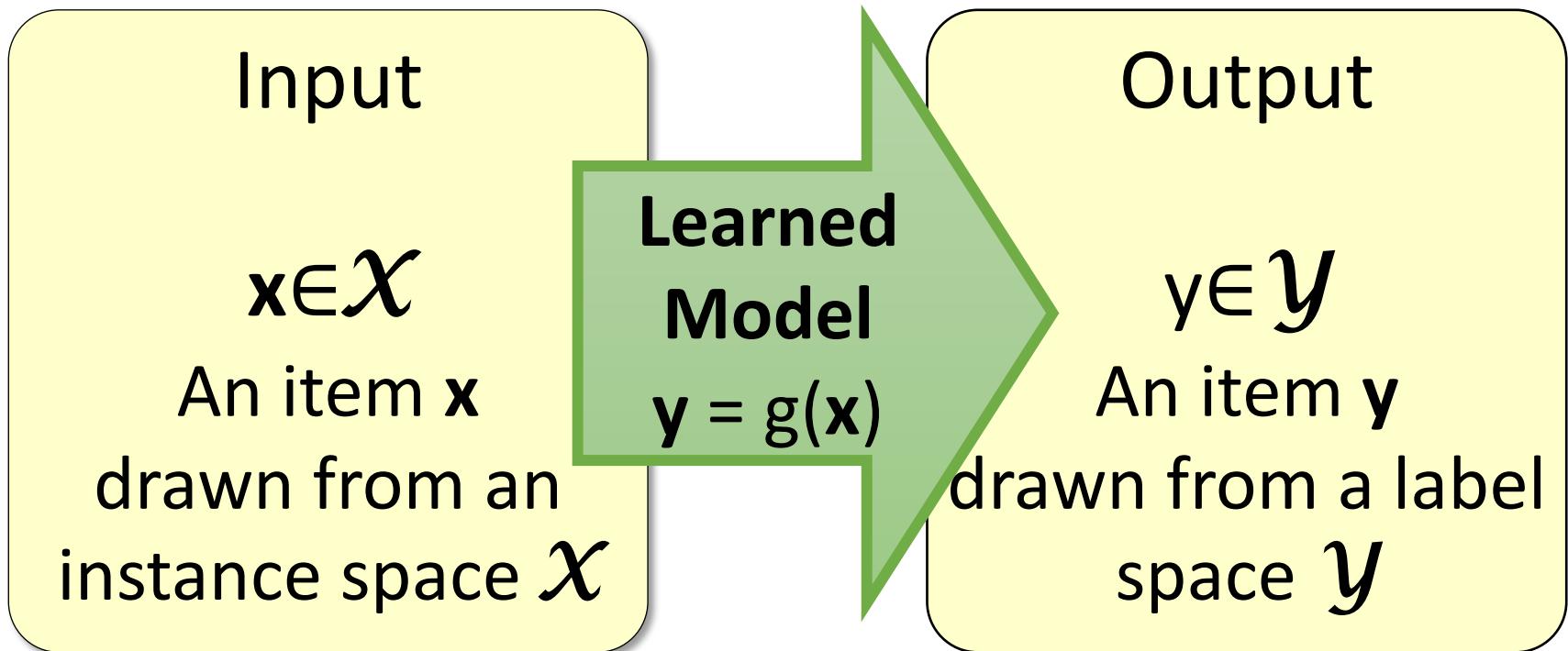
Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, Jessica Wu (Harvey Mudd), David Kauchak (Pomona), Dan Roth (Upenn), Sriram Sankararaman (UCLA), whose slides are also heavily used, and the many others who made their course materials freely available online.

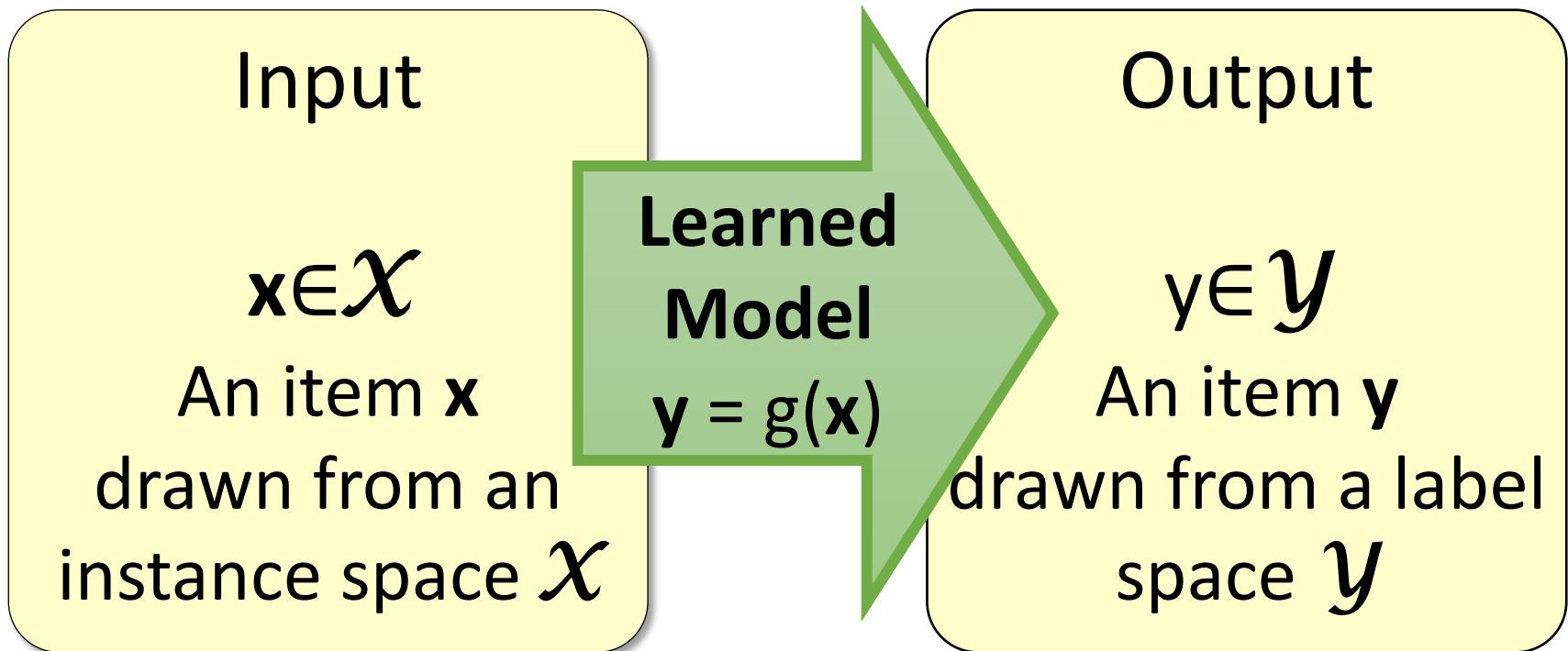
Supervised Learning



Using supervised learning

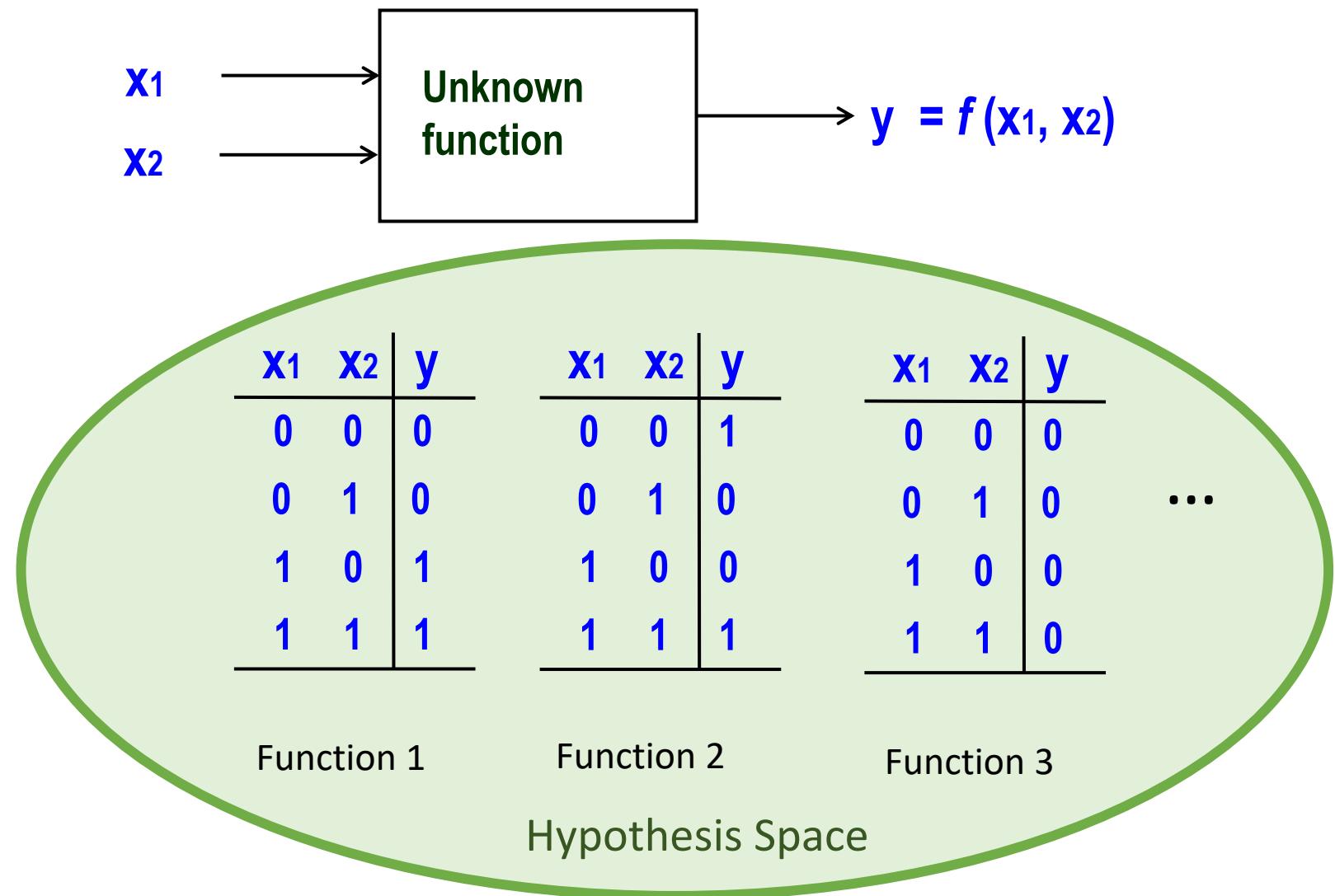
- ❖ What is our instance space?
 - ❖ Gloss: What kind of features are we using?
- ❖ What is our label space?
 - ❖ Gloss: What kind of learning task are we dealing with?
- ❖ What is our hypothesis space?
 - ❖ Gloss: What kind of functions (models) are we learning?
- ❖ What learning algorithm do we use?
 - ❖ Gloss: How do we learn the model from the labeled data?
- ❖ What is our loss function/evaluation metric?
 - ❖ Gloss: How do we measure success? What drives learning?

3. The model $g(\mathbf{x})$

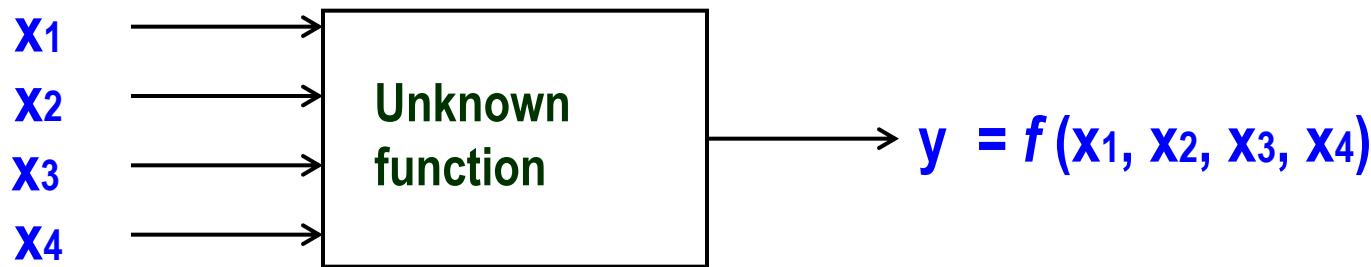


- ❖ We need to choose what *kind* of model we want to learn

Boolean Function



A Learning Problem



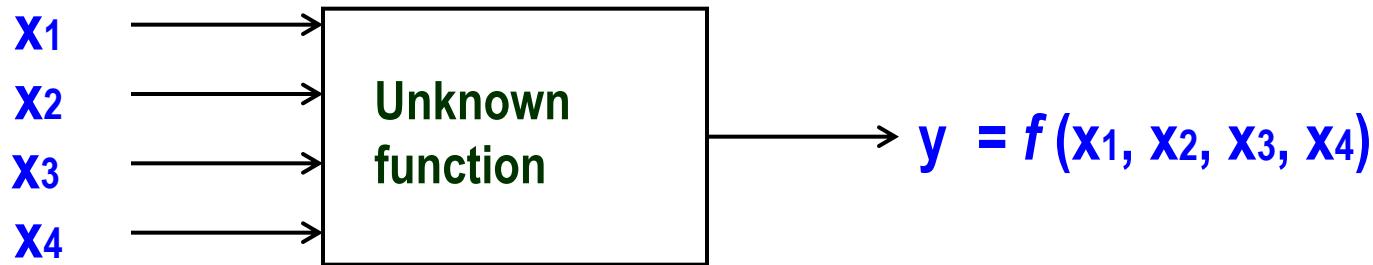
Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

A function g is consistent to a dataset
 $D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$

Discussion: A Learning Problem



Example	x_1	x_2	x_3	x_4	y	
1	0	0	1	0	0	Can you learn this function?
2	0	1	0	0	0	What is it?
3	0	0	1	1	1	A function g is consistent to a dataset
4	1	0	0	1	1	$D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$
5	0	1	1	0	0	
6	1	1	0	0	0	How many possible functions over four features?
7	0	1	0	1	0	How many function is consistent to D on the left

Hypothesis Space

How many possible functions over four features?

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

Example	x1	x2	x3	x4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	0	1	0	?
0	0	0	1	1	?
0	1	0	0	0	?
0	1	0	0	1	?
0	1	1	0	0	?
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	?
1	0	1	0	0	?
1	0	1	1	0	?
1	1	0	0	0	?
1	1	0	0	1	?
1	1	1	0	0	?
1	1	1	1	0	?
1	1	1	1	1	?

Hypothesis Space

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Is Learning Possible?

which one is the most likely one?

Example	X1	X2	X3	X4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	0
	0	1	1	1	?
	1	0	0	0	?
	1	0	0	1	1
	1	0	1	0	?
	1	0	1	1	?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

We can't
corre-
pos-

After

have 2^T possibilities for T

- There are $|Y|^{|X|}$ possible functions $f(x)$ from the instance space X to the label space Y .

- Learners typically consider **only a subset** of the functions from X to Y , called the hypothesis space H . $H \subseteq |Y|^{|X|}$

Is Learning Possible?

Example	X1	X2	X3	X4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
					1
					0
					0
					?
					1
					0
					0
					?
					1
					?
					?
					1
					?
					?
					0
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space (2)

Simple Rules: **conjunctive rules**

of the form $y = x_i \wedge x_j \wedge \dots \wedge x_k$

e.g., $y = x_2 \wedge x_3$

$y = x_1 \wedge x_2 \wedge X_4$

How large is the hypothesis space?

Hypothesis Space (2)

1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	0	1	1	1	1
4	1	0	0	0	1	1
5	1	0	1	1	0	0
6	1	1	0	0	0	0
7	0	1	0	1	0	0

Simple Rules: There are only 16 **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=1$		$x_2 \wedge x_3$	
x_1		$x_2 \wedge x_4$	
x_2		$x_3 \wedge x_4$	
x_3		$x_1 \wedge x_2 \wedge x_3$	
x_4		$x_1 \wedge x_2 \wedge x_4$	
$x_1 \wedge x_2$		$x_1 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_3$		$x_2 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_4$		$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	

Hypothesis Space (2)

1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	0	1	1	1	1
4	1	0	0	1	1	1
5	1	0	1	1	0	0
6	1	1	0	0	0	0
7	0	1	0	1	0	0

Simple Rules: There are only 16 **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=c$		$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
x_2	0100 0	$x_3 \wedge x_4$	1001 1
x_3	0110 0	$x_1 \wedge x_2 \wedge x_3$	0011 1
x_4	0101 1	$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_2$	1100 0	$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

No simple rule explains the data. The same is true for **simple clauses**.

Hypothesis Space (3)

m-of-n rules: There are 32 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>	<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>
{X ₁ }					{X ₂ , X ₄ }				
{X ₂ }					{X ₃ , X ₄ }				
{X ₃ }					{X ₁ , X ₂ , X ₃ }				
{X ₄ }					{X ₁ , X ₂ , X ₄ }				
{X ₁ , X ₂ }					{X ₁ , X ₃ , X ₄ }				
{X ₁ , X ₃ }					{X ₂ , X ₃ , X ₄ }				
{X ₁ , X ₄ }					{X ₁ , X ₂ , X ₃ , X ₄ }				
{X ₂ , X ₃ }									

Hypothesis Space (3)

m-of-n rules: There are 32 possible rules of the form " $y = 1$ if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

variables	1-of	2-of	3-of	4-of	variables	1-of	2-of	3-of	4-of
{X1}	3	-	-	-	{X2, X4}	2	3	-	-
{X2}	2	-	-	-	{X3, X4}	4	4	-	-
{X3}	1	-	-	-	{X1, X2, X3}	1	3	3	-
{X4}	7	-	-	-	{X1, X2, X4}	2	3	3	-
{X1, X2}	2	3	-	-	{X1, X3, X4}	1	***	3	-
{X1, X3}	1	3	-	-	{X2, X3, X4}	1	5	3	-
{X1, X4}	6	3	-	-	{X1, X2, X3, X4}	1	5	3	3
{X2, X3}	2	3	-	-					

Found a consistent hypothesis.
15 Lec 3: Model & KNN

Views of Learning

- ❖ Learning is the removal of our remaining uncertainty:
- ❖ Learning requires guessing a good hypothesis class
 - ❖ Start with a small class and enlarge it until it contains an hypothesis that fits the data.
- ❖ We could be wrong !
 - ❖ Our guess of the hypothesis space could be wrong
 - ❖ $y=x4 \wedge$ one-of (x_1, x_3) is also consistent

General strategies for Machine Learning

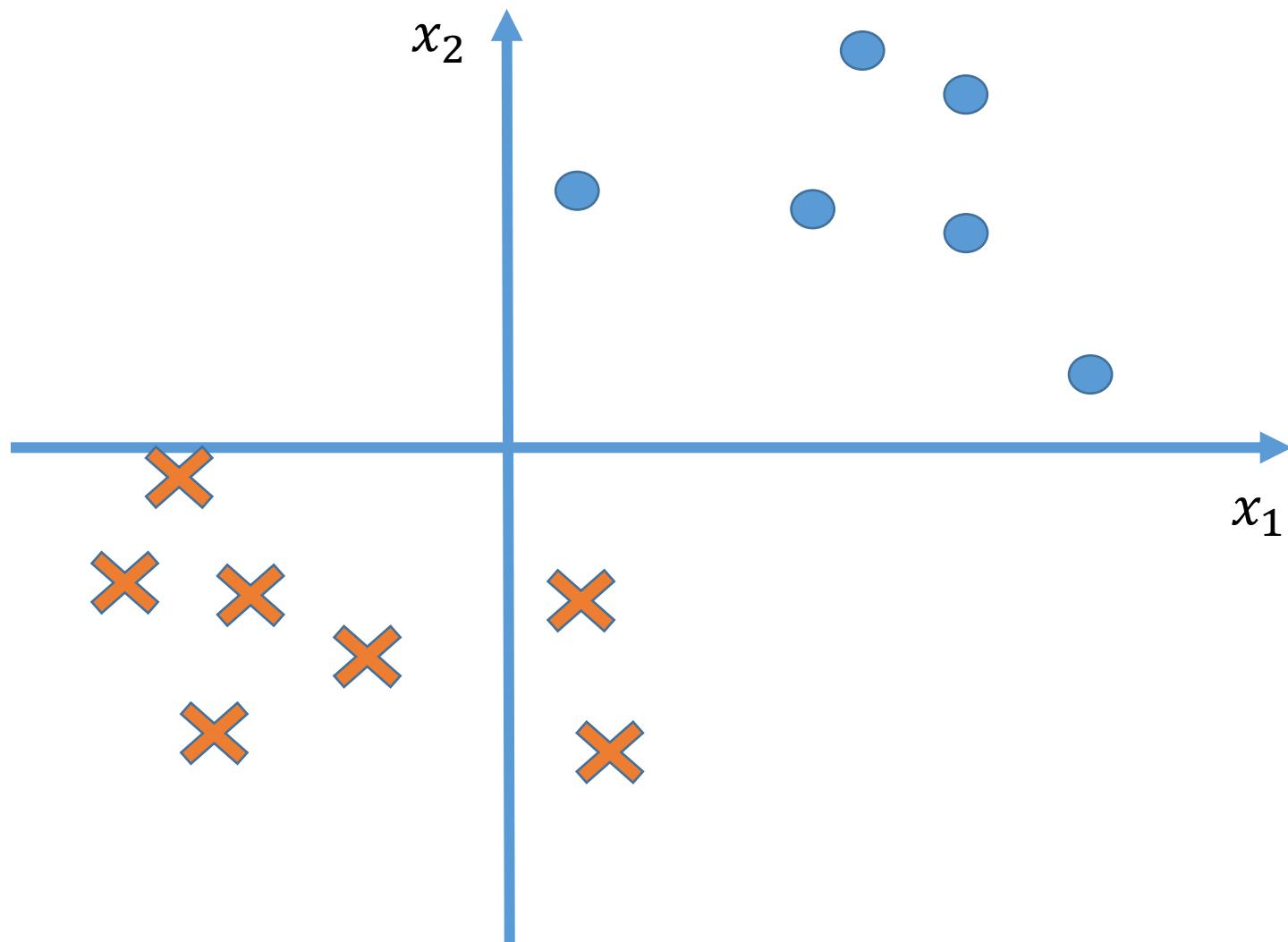
- ❖ Develop flexible hypothesis spaces:
 - ❖ Decision trees, neural networks, nested collections.
- ❖ Develop algorithms for finding the "best" hypothesis in the hypothesis space, that fits the data
- ❖ And, hope that it will generalize well

Hypothesis Space -- Real-Value Features

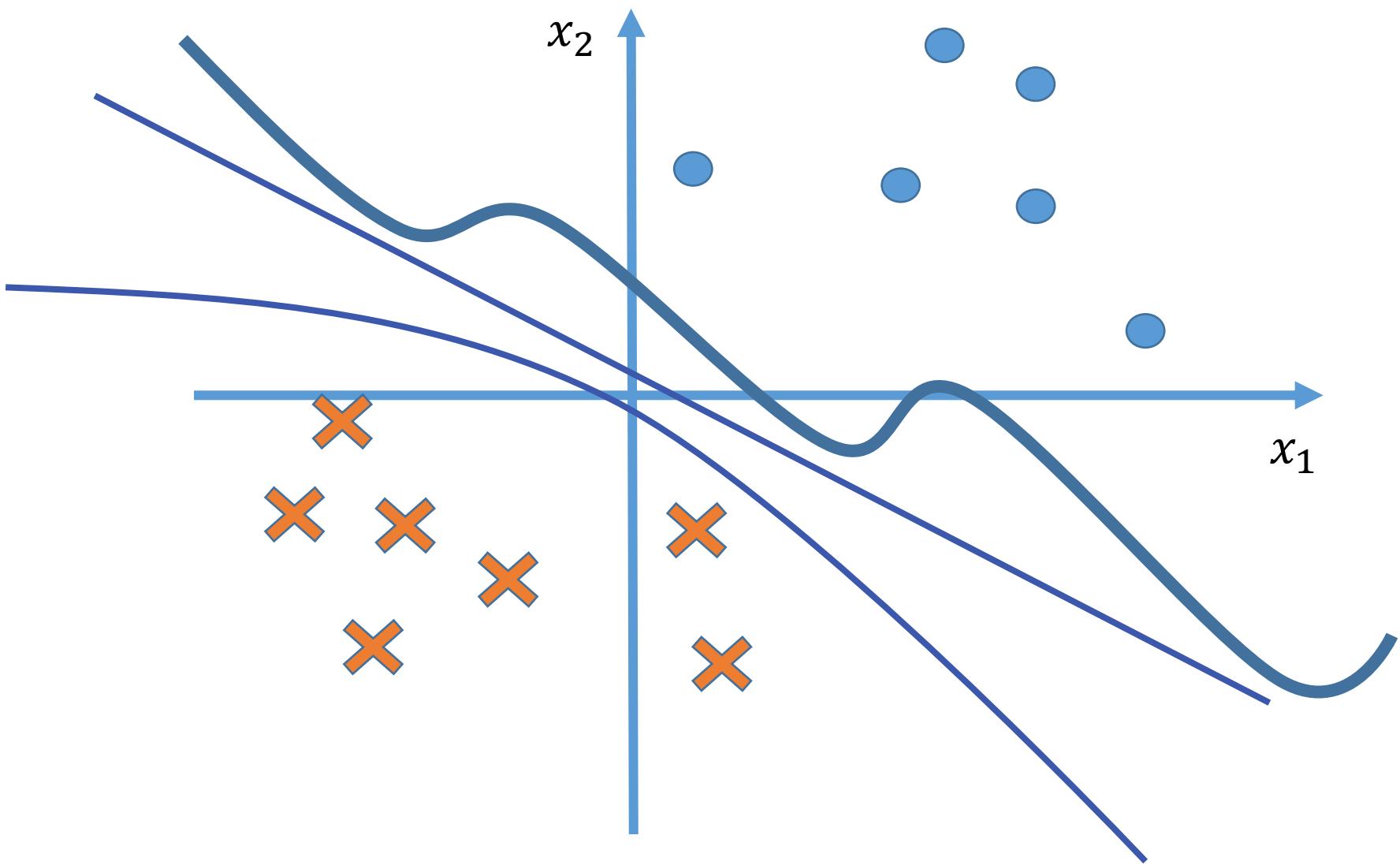
WHAAAAA?!?!



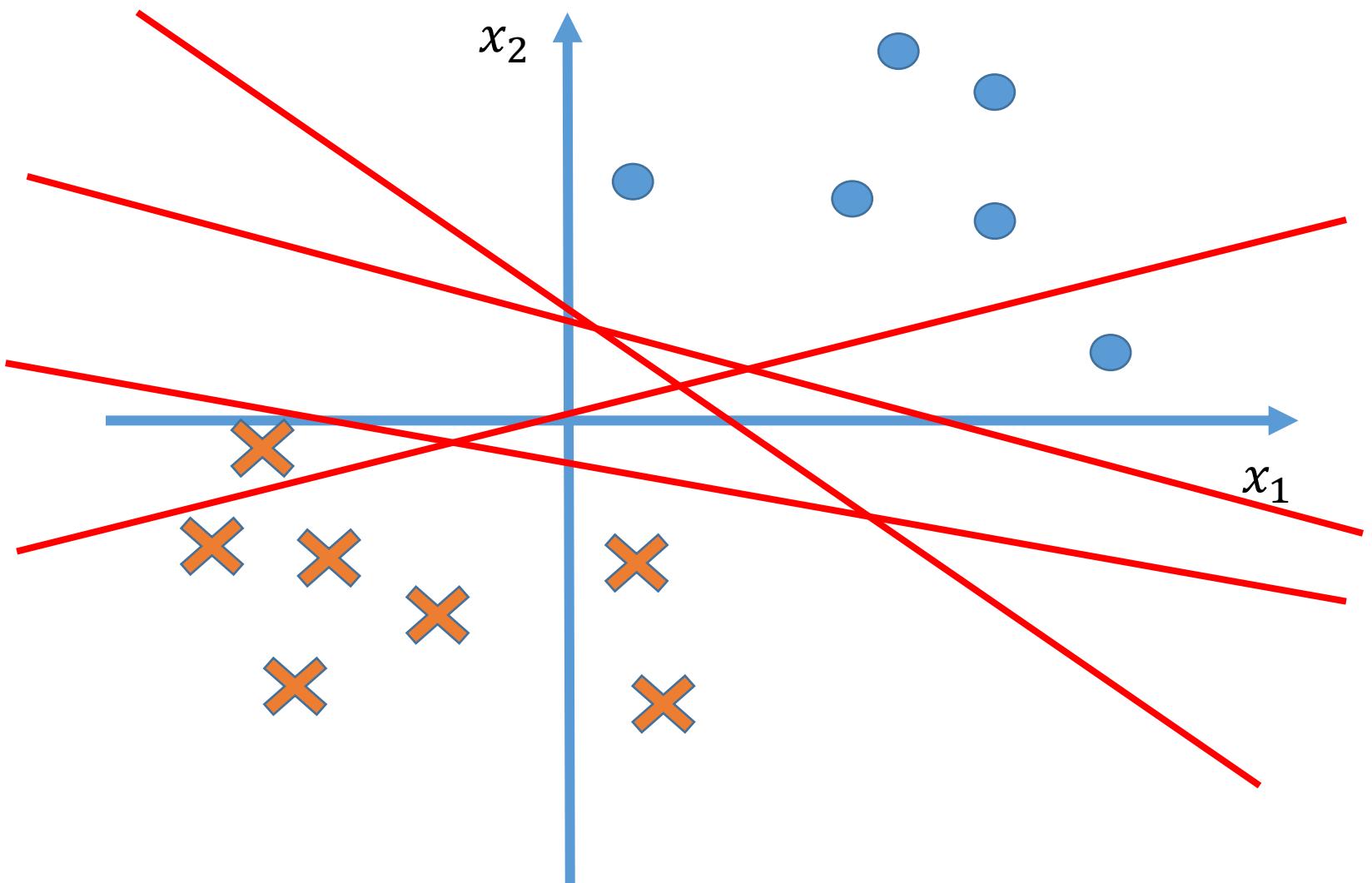
Example problem



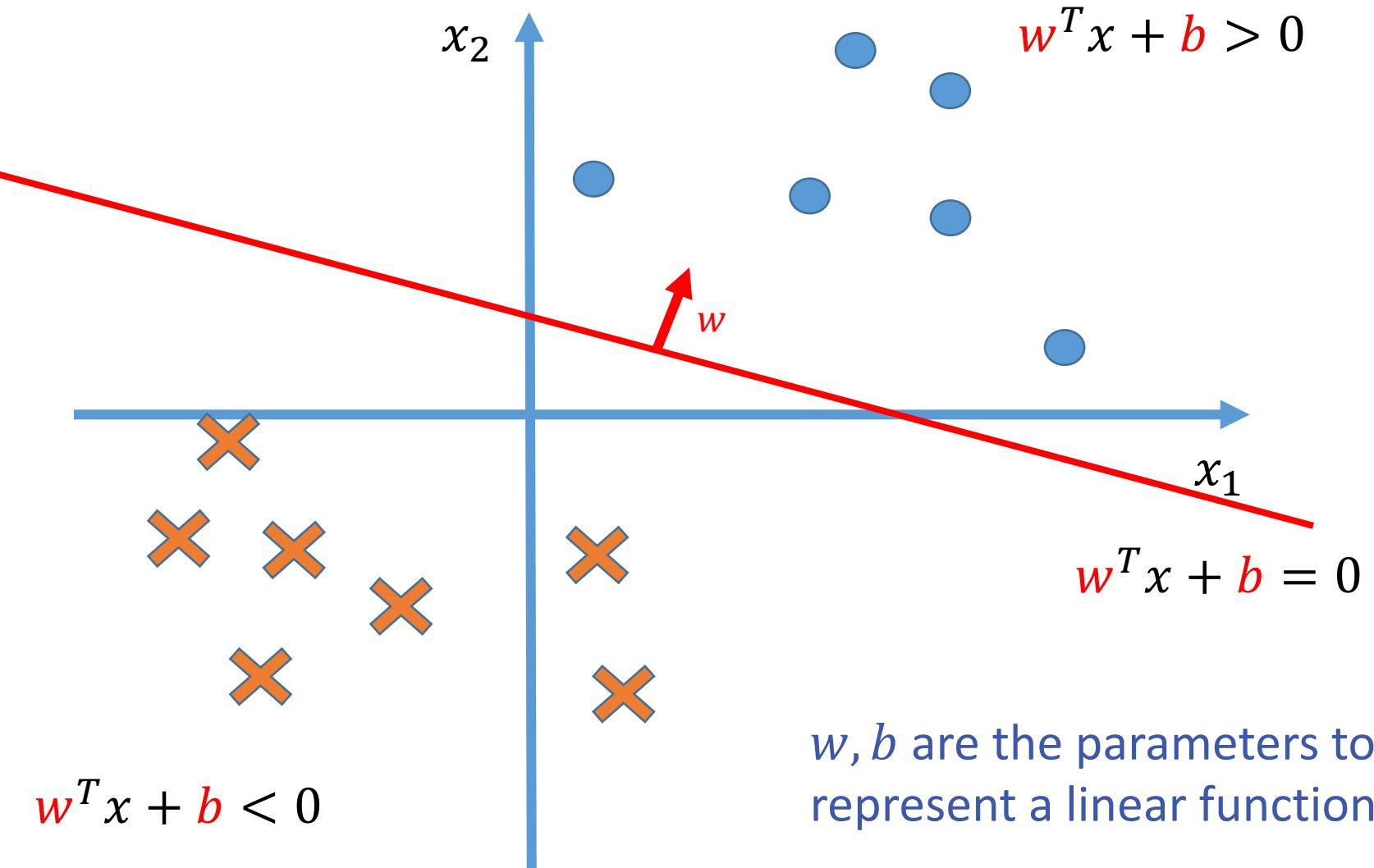
Hypothesis space:



Hypothesis space: linear model



Hypothesis space: linear model



How to learn?

How can we find a good model from the hypothesis space?

Recap: rule-out

m-of-n rules: There are 32 possible rules of the form " $y = 1$ if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

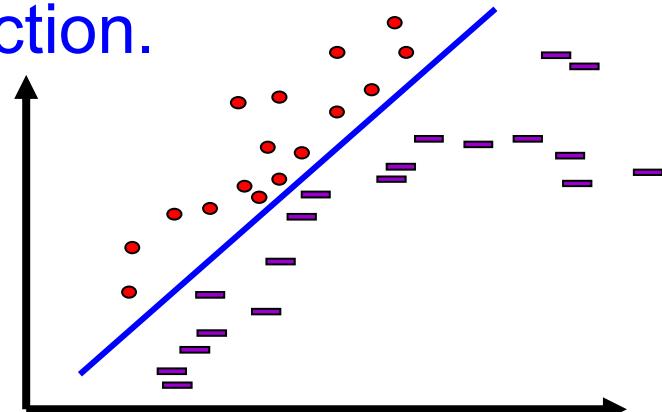
Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

variables	1-of	2-of	3-of	4-of	variables	1-of	2-of	3-of	4-of
{X1}									
{X2}									
{X3}									
{X4}									
{X1,X2}									
{X1, X3}									
{X1, X4}									
{X2,X3}	2	3	-	-					

Learning is the removal of our
remaining uncertainty

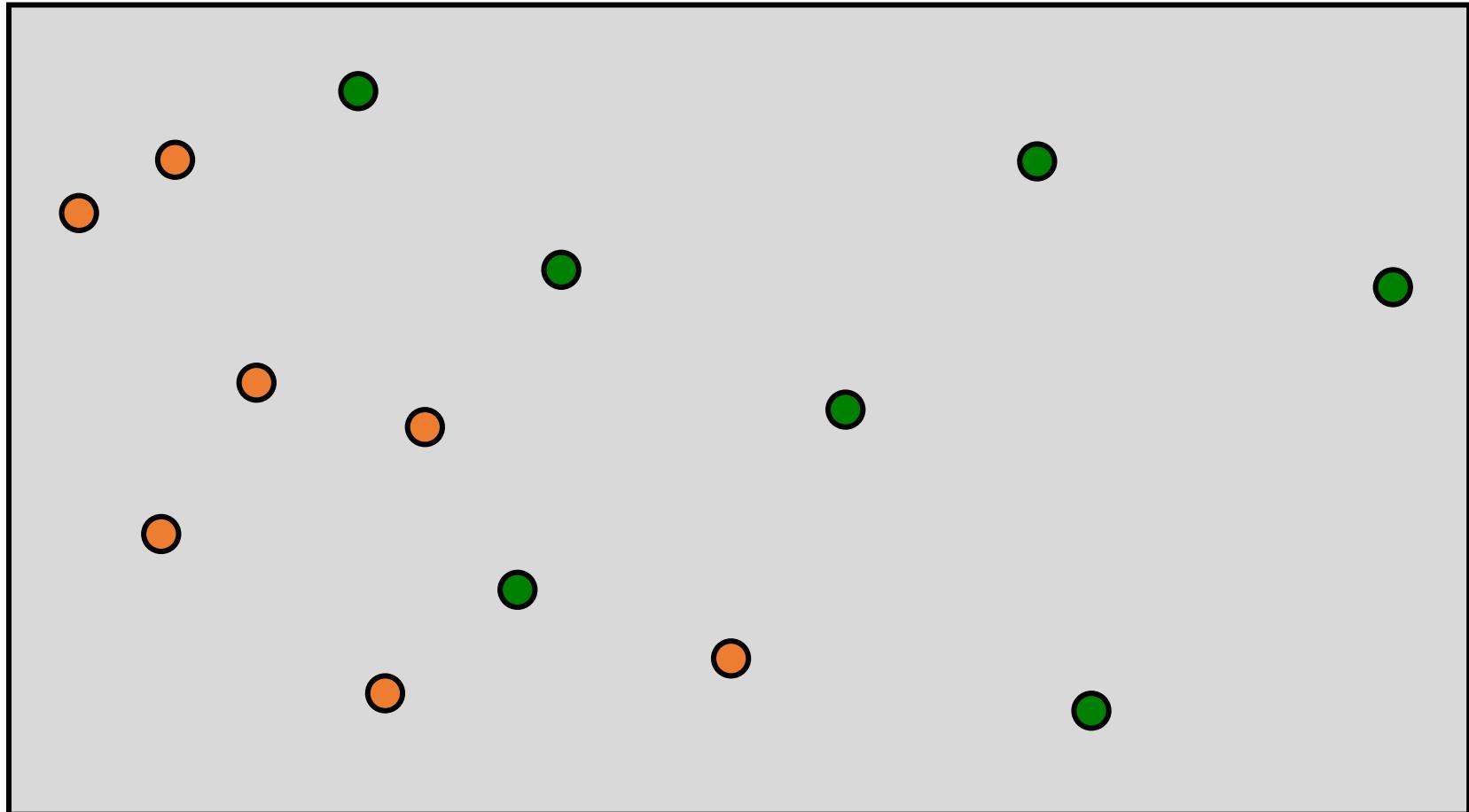
How about linear function?

- ❖ Challenges
 - ❖ The hypothesis space contains infinite # functions
 - ❖ Several functions are consistent with the data
- ❖ A possibility: Local search
 - ❖ Start with a linear threshold function.
 - ❖ See how well you are doing.
 - ❖ Correct
 - ❖ Repeat until you converge.
- ❖ Optimize a function with calculus

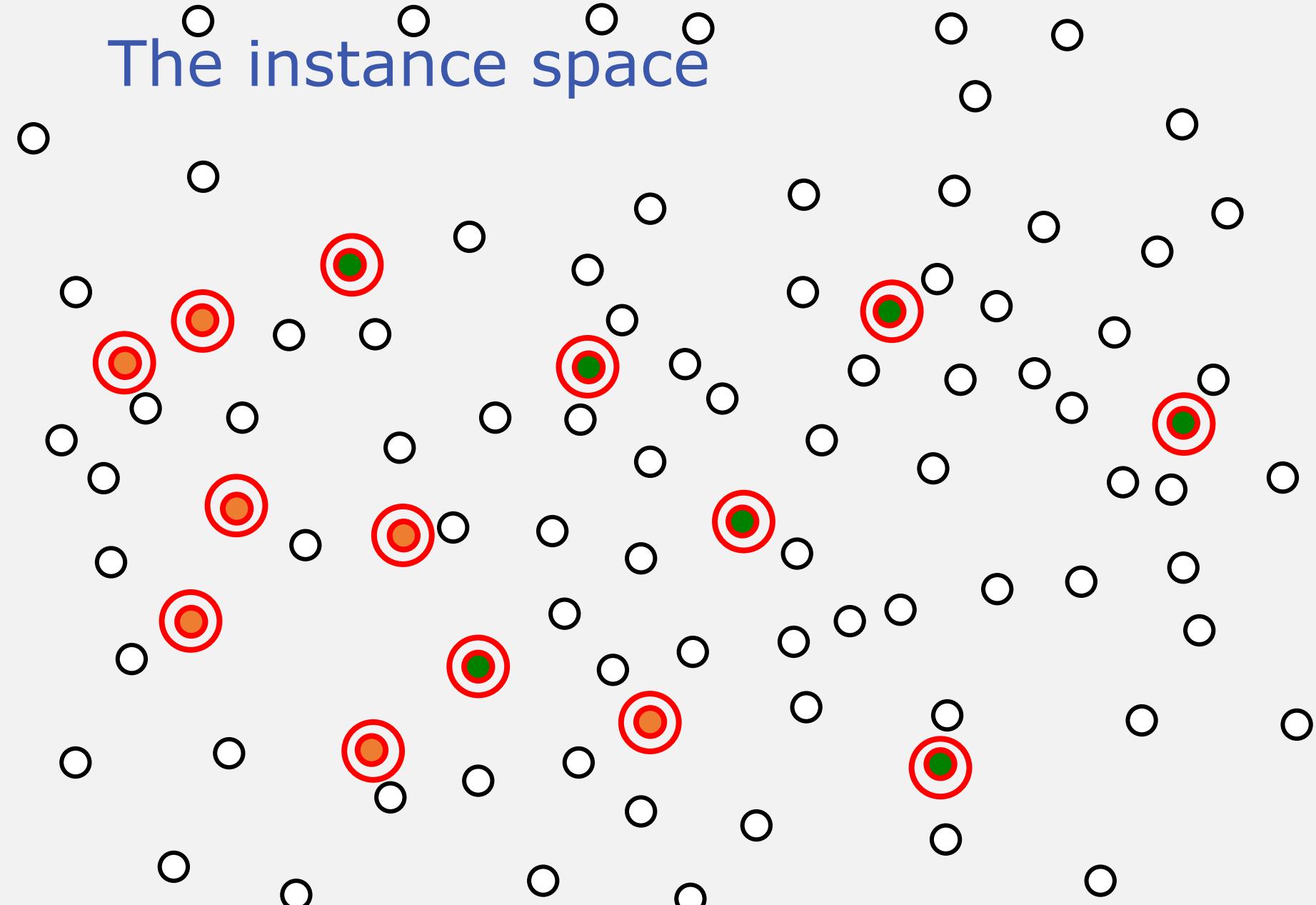


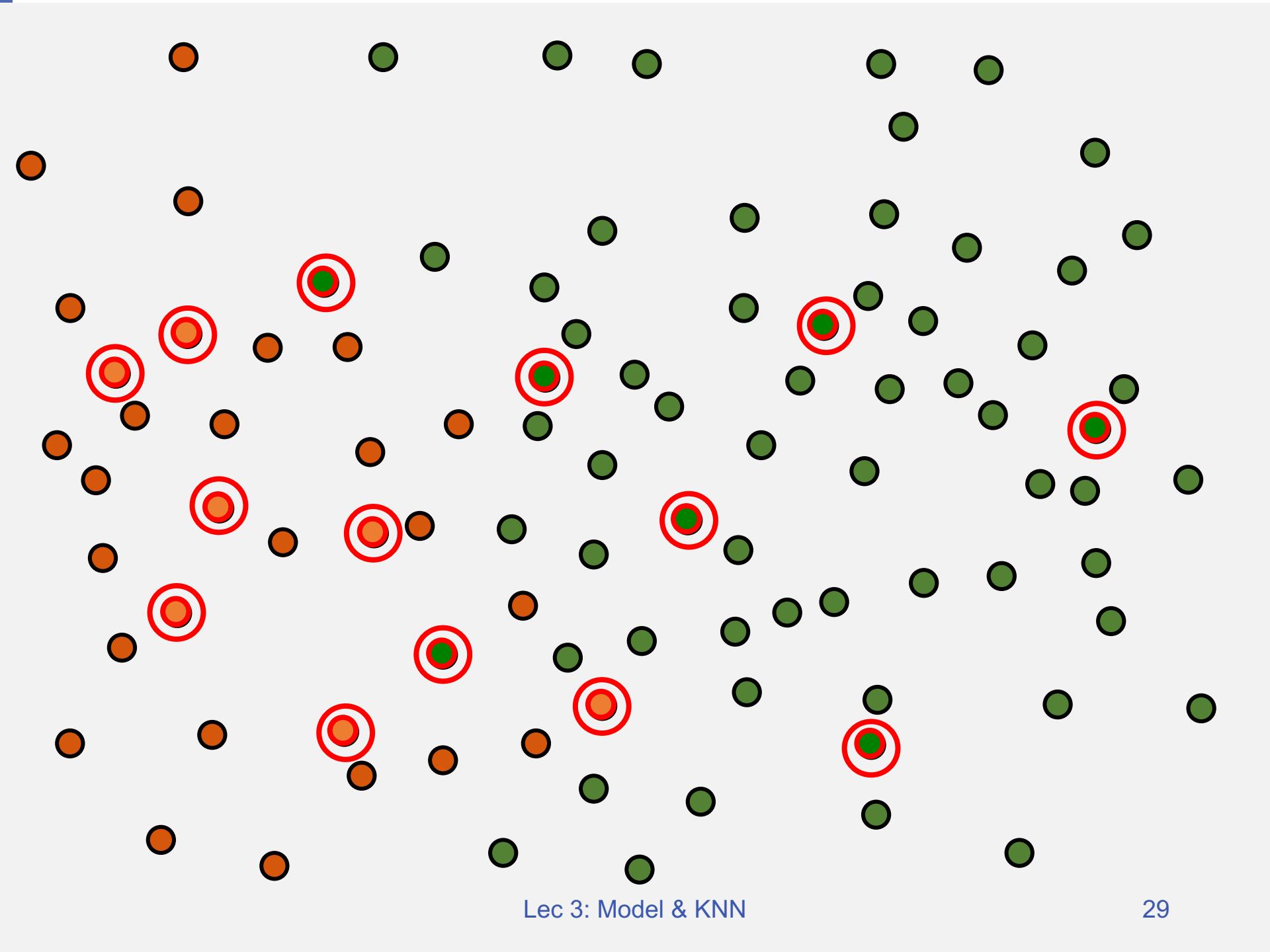
Choose Hypothesis Space

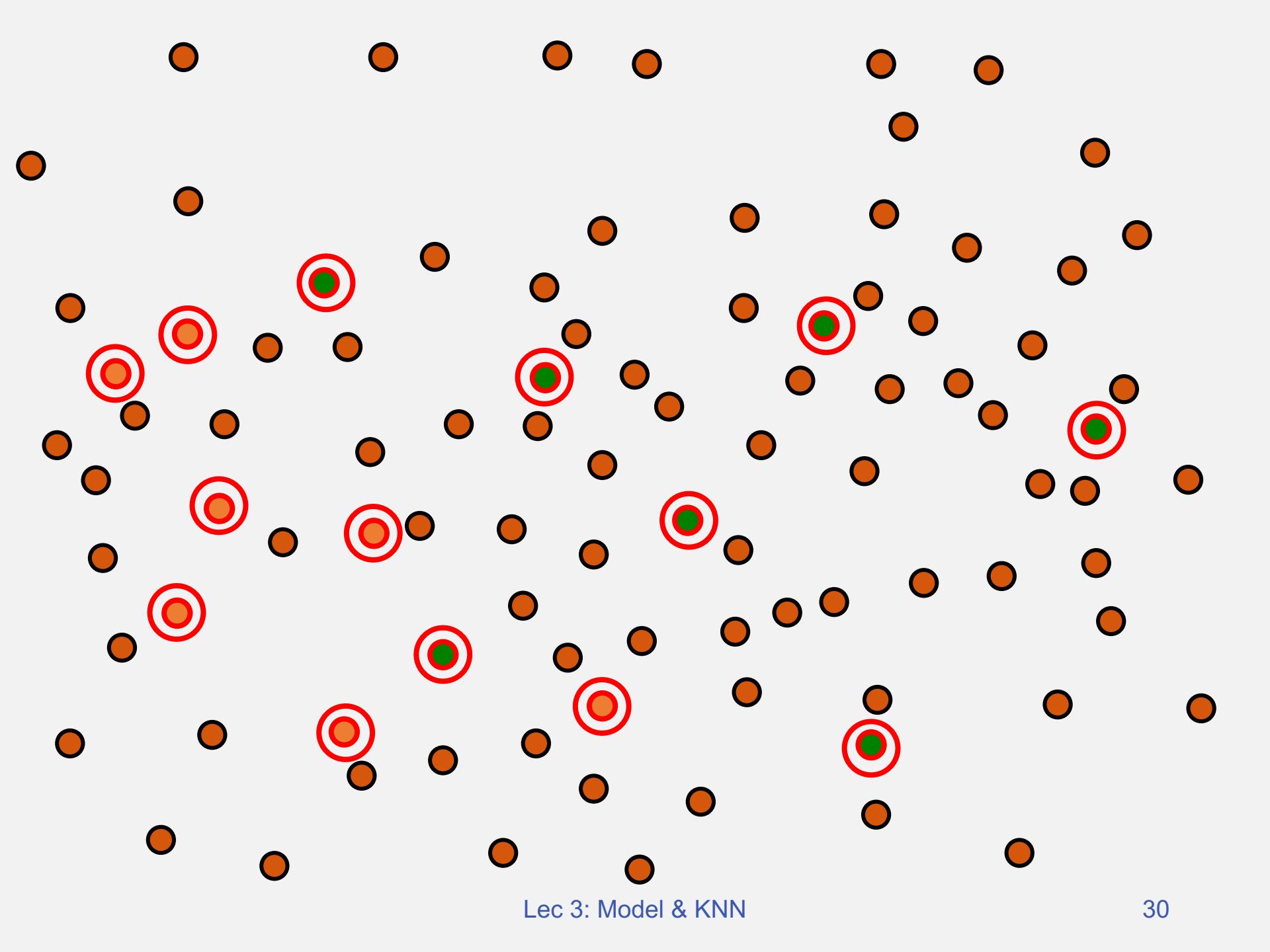
Our training data



The instance space

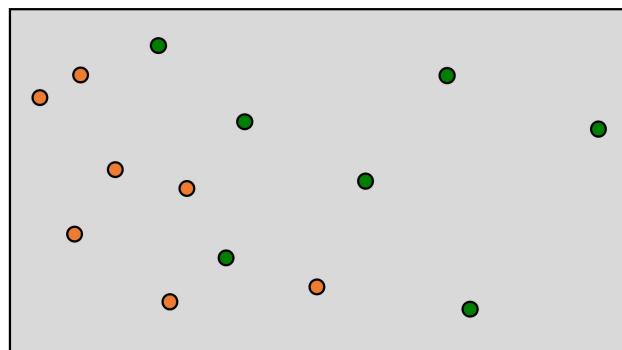




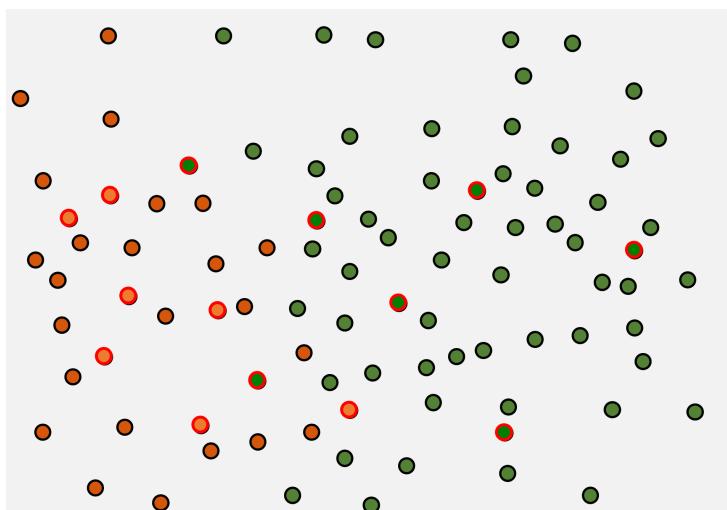


Which one is more likely?

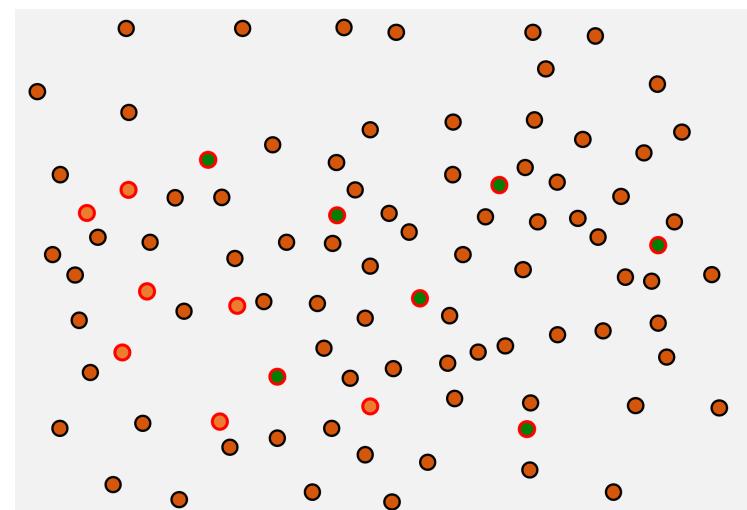
Training set



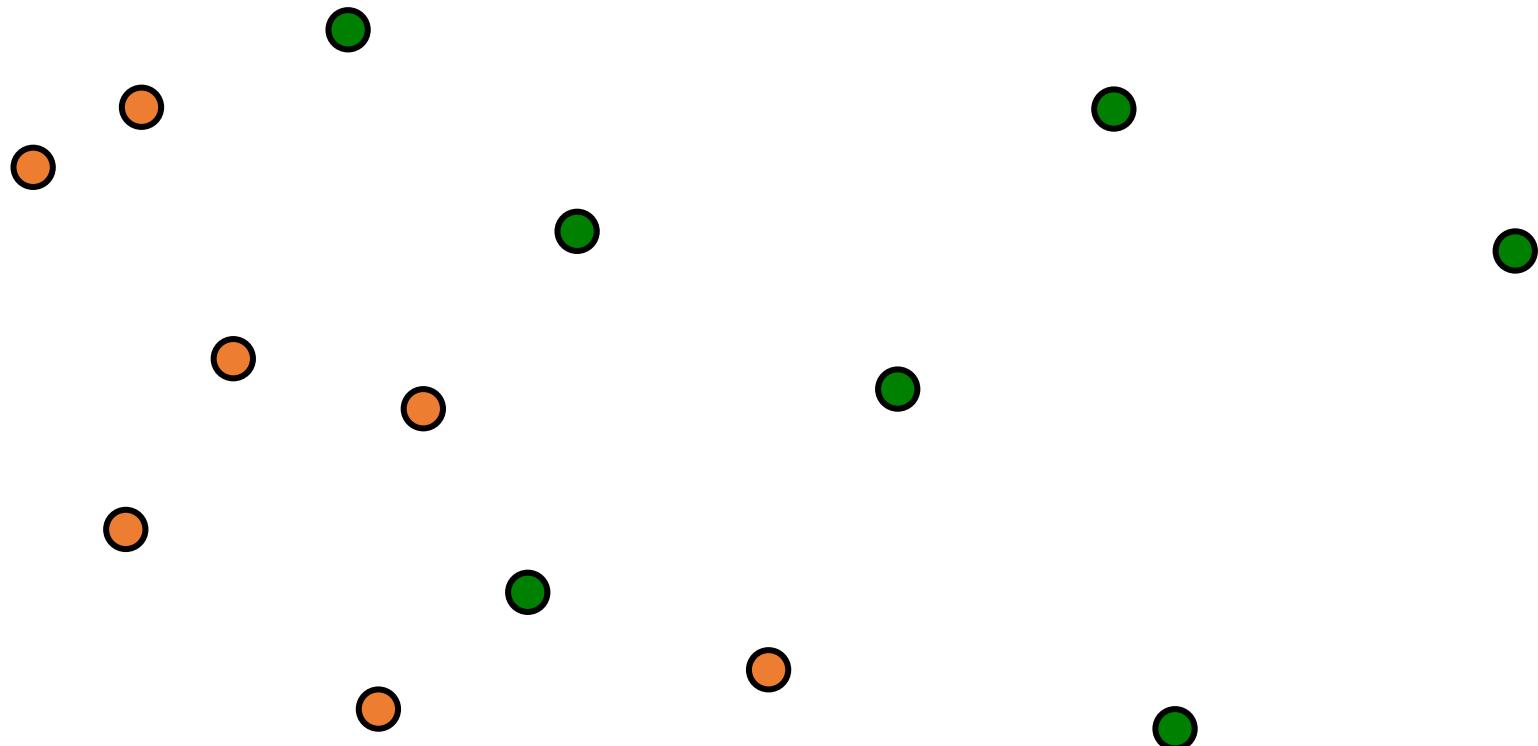
Data distribution A



Data distribution B

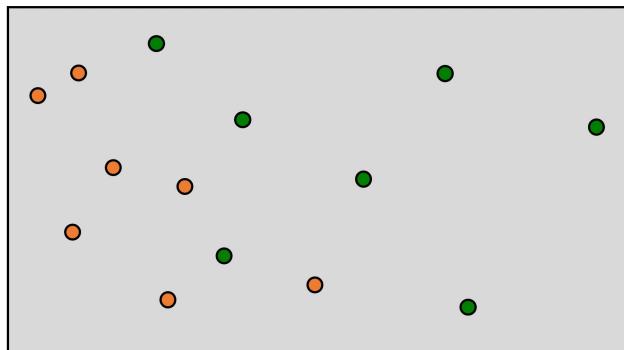


Our training data

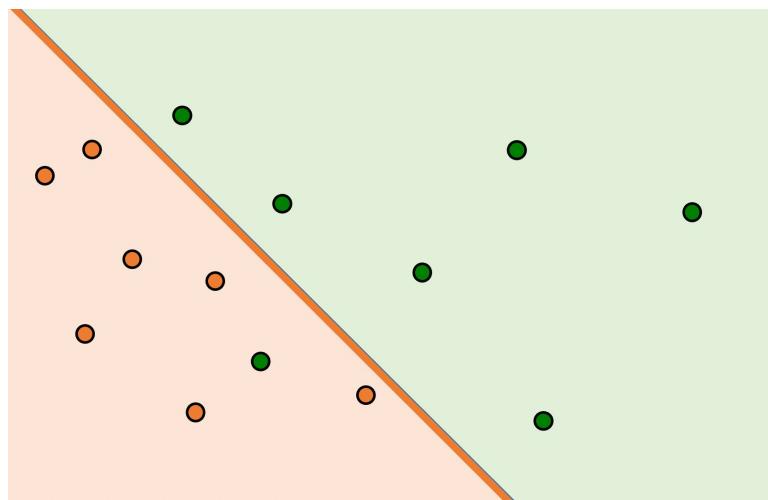


Which one is more likely to be a good hypothesis?

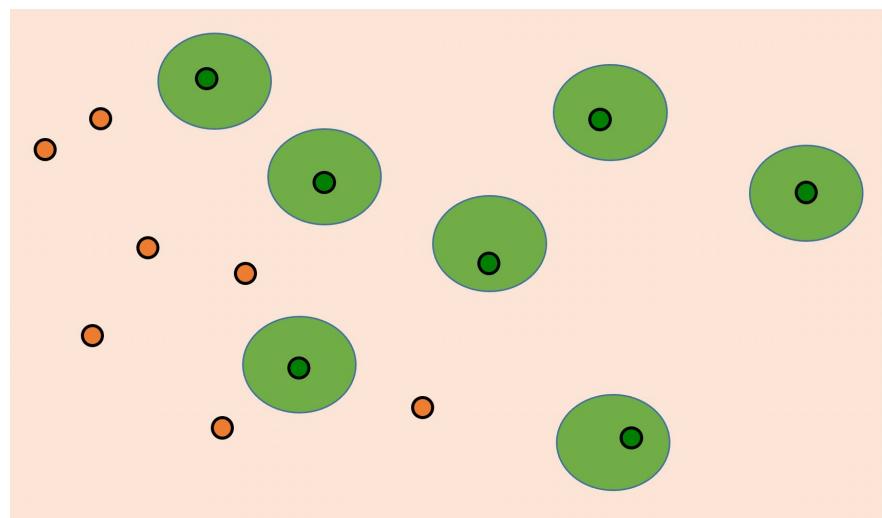
Training set



Hypothesis A

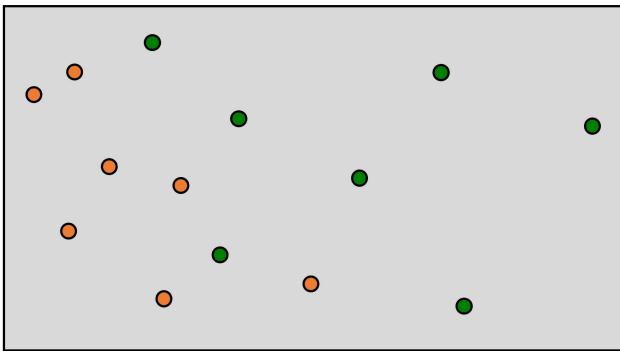


Hypothesis B

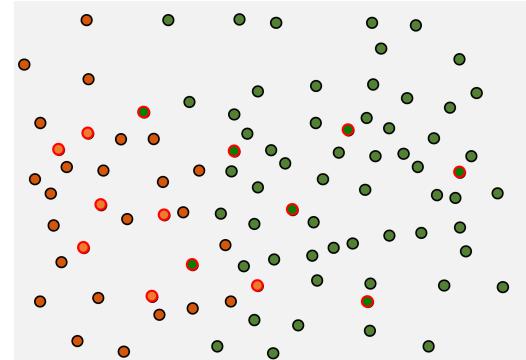


Which one is more likely to be a good hypothesis?

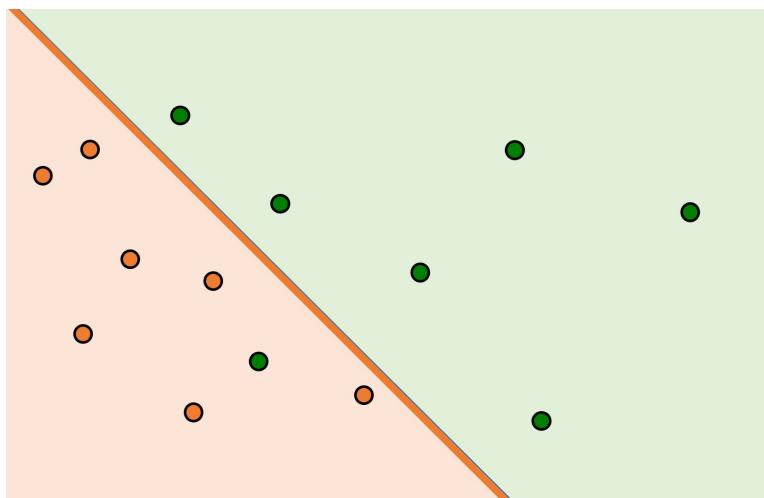
Training set



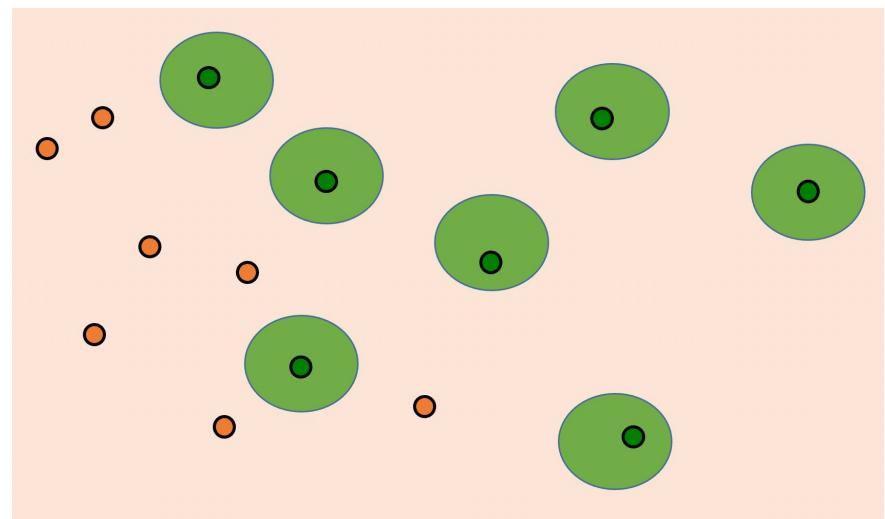
(Likely) Data Distribution



Hypothesis A

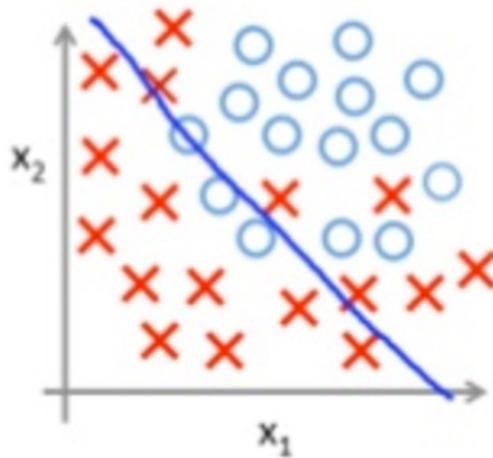


Hypothesis B

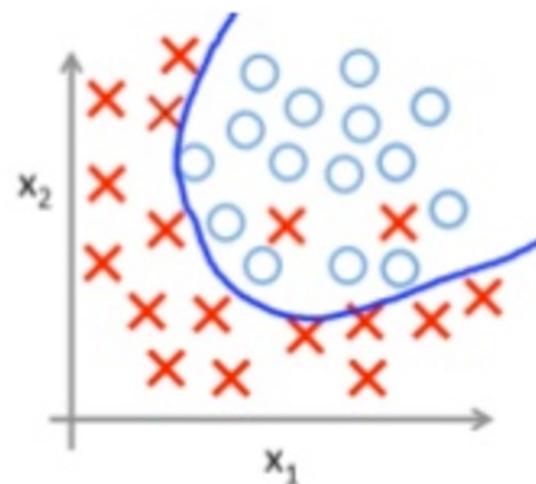


Under-fitting and over-fitting

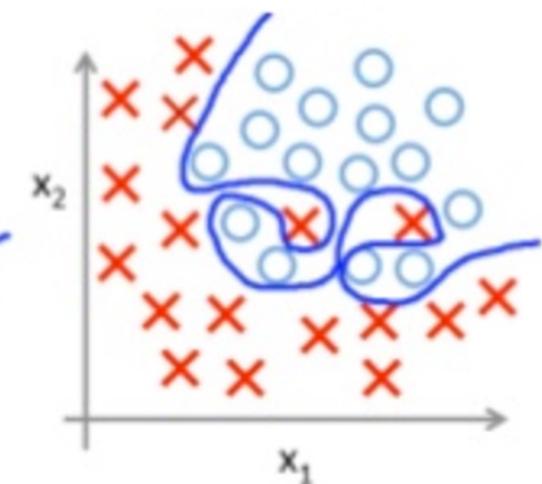
- ❖ Which classifier (blue line) is the best one?



(A)



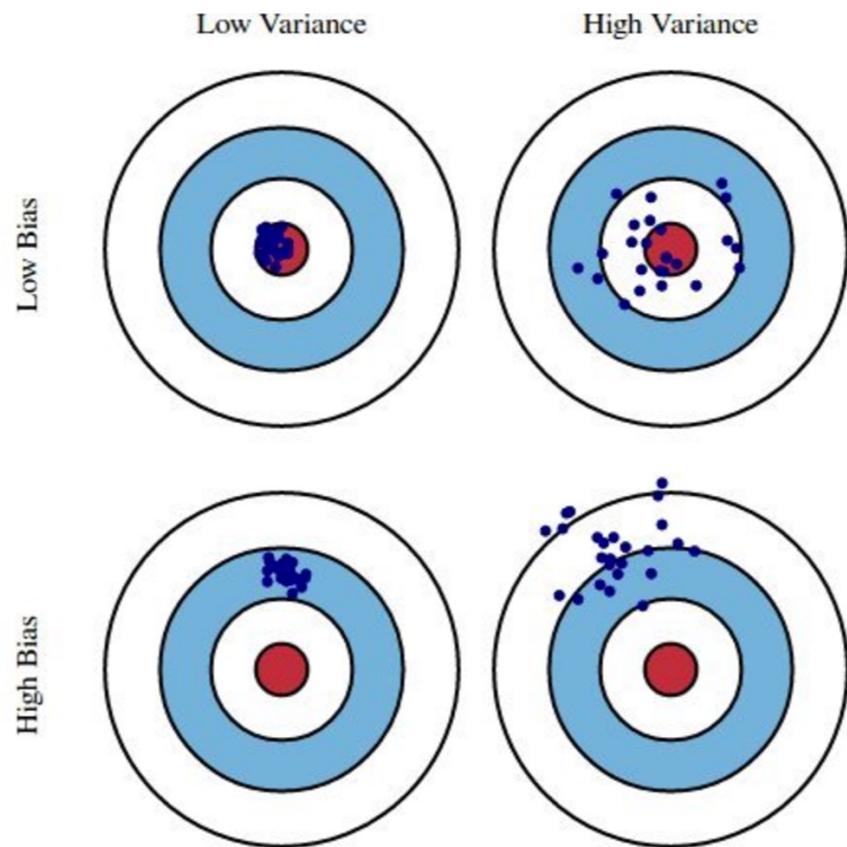
(B)



(C)

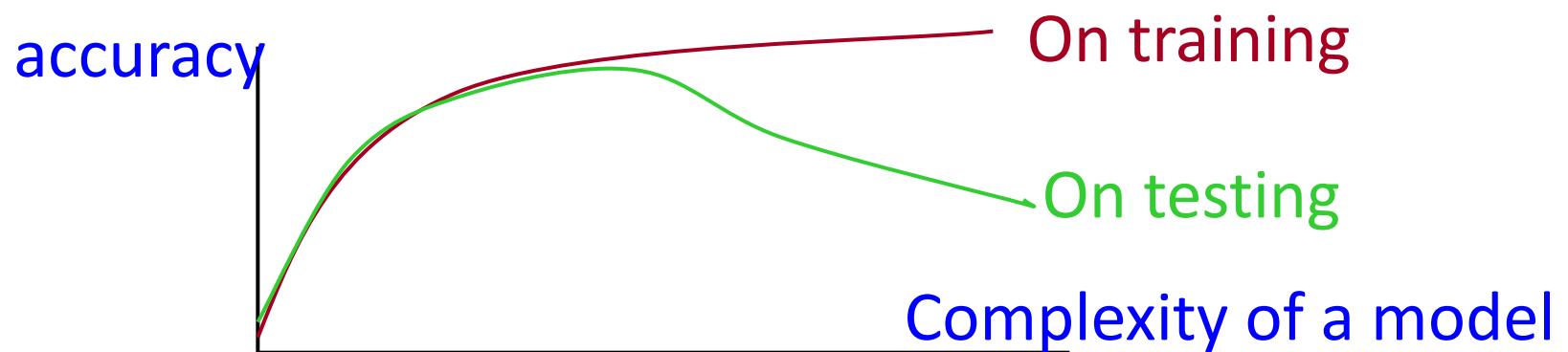
Bias V.S. Variance

- ❖ Remember, training data are subsamples drawn from the true distribution
- ❖ Exam strategy:
 - ❖ Study every chapter well
 - ❖ A+: Low var & bias
 - ❖ Study only a few chapters
 - ❖ A+? B? C? Low bias; High var
 - ❖ Study every chapter roughly
 - ❖ B+: Low var; high bias
 - ❖ Go to sleep
 - ❖ B ~D: High var, high bias



Overfitting the Data

- ❖ A classifier perform perfectly on the training data may not lead to the best generalization performance.
 - ❖ There may be noise in the training data
 - ❖ The algorithm might be making decisions based on very little data



Prevent overfitting

- ❖ Using a less-expressive model
 - ❖ E.g., linear model
- ❖ Adding regularization
 - ❖ Promote simpler models
- ❖ Data perturbation (add noise in training)
 - ❖ Can be done algorithmically (e.g., dropout)
- ❖ Stop the optimization process earlier
 - ❖ Sounds bad in theory; but works in practice

More discussion in later lectures

K-Nearest Neighbor

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and many others who made their course material freely available online.

Motivation: spam mail

hi Spam x

! marina <marina0279@static.vnpt.vn> Jan 16 (7 days ago) ★
to kw ↗

This message has a from address in static.vnpt.vn but has failed static.vnpt.vn's required tests for authentication. [Learn more](#)

You seem like my type and I would like to know you more! Write me if you are interested, here is my email denisavafursula@rambler.ru and, if you want, I will send some of my photos.
Hugs, marina

<input type="checkbox"/>	★ Victoria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email vanesammrenate@	Jan 17
<input type="checkbox"/>	★ Oksana	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elenarzpilse@rambl	Jan 16
<input type="checkbox"/>	★ Oksana	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venhelgarjxq@raml	Jan 16
<input type="checkbox"/>	★ Victoria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elipetrayk@rambler	Jan 16
<input type="checkbox"/>	★ Natasha	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email lucietmsabine@rar	Jan 16
<input type="checkbox"/>	★ Daria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email rokcmingrid@rambl	Jan 16
<input type="checkbox"/>	★ Katya	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email michaelavat62@rar	Jan 16
<input type="checkbox"/>	★ Yulia	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elisa4inge@ramble	Jan 16
<input type="checkbox"/>	★ Veronika	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venlsajutta@ramble	Jan 16
<input type="checkbox"/>	★ Tanya	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email ellairxt6t@rambler.r	Jan 16

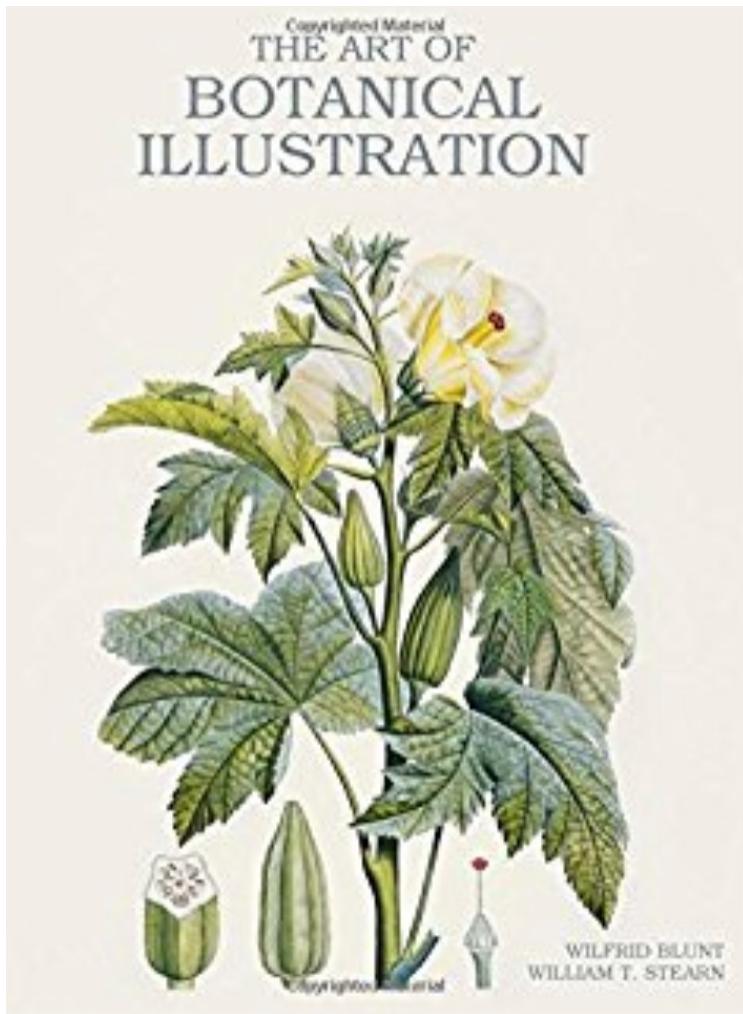
Motivation: Learning from memorization

Recognizing flowers

- ❖ What is this flower called?



Look up at Botanical Illustration Books



Motivation: Learning from memorization

Recognizing flowers

Types of Iris: **setosa, versicolor, and virginica**



(A)



(B)



(C)



Lec 3. Model & KNN

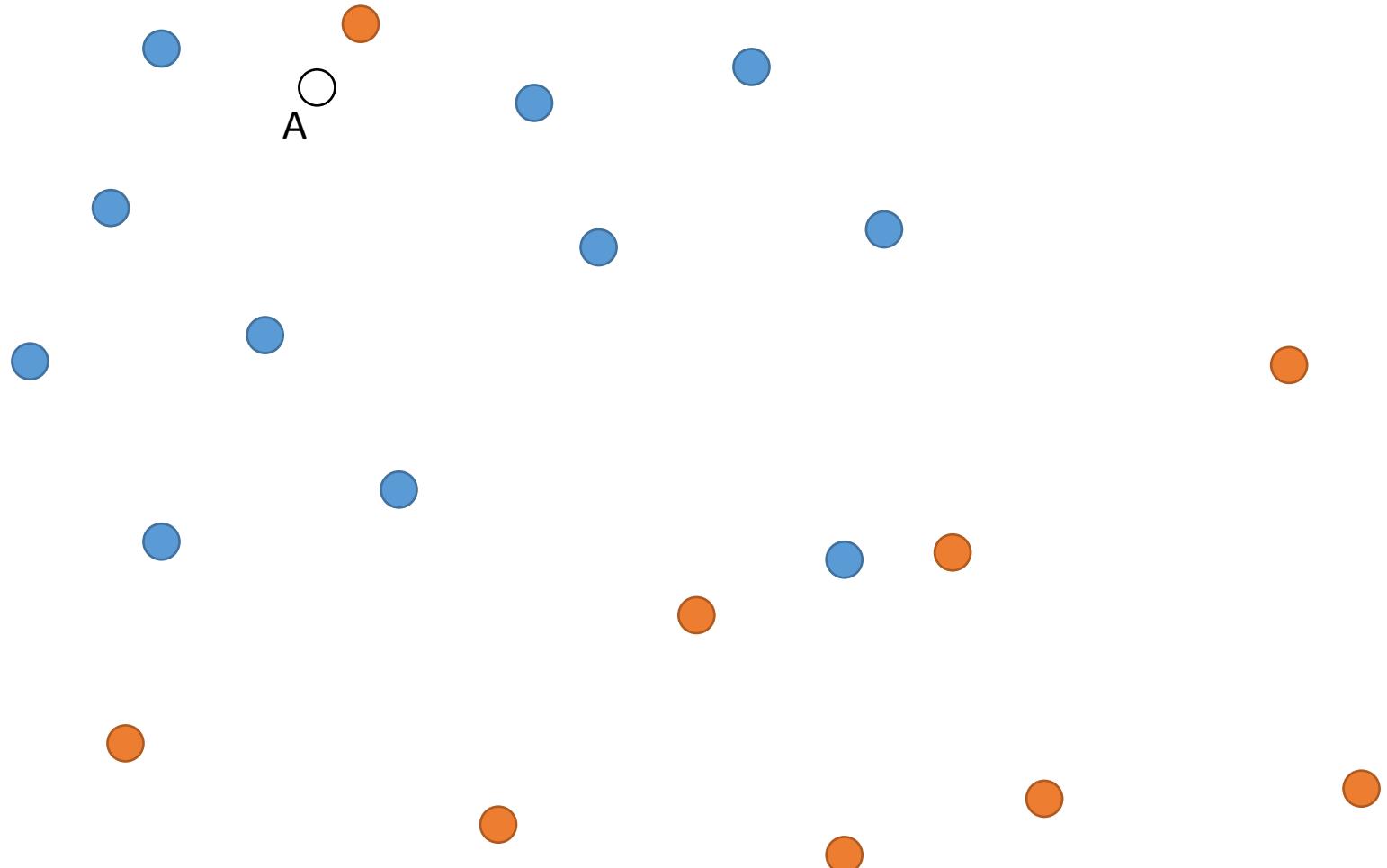
What you will learn in this lecture

- ❖ k-NN algorithm
- ❖ Distance between in the vector space
- ❖ Parameter tuning
- ❖ Decision boundary
- ❖ Curse of the Dimensionality
(Theoretical Limitation)

Nearest Neighbors: The basic version

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction: for a new example \mathbf{x}
 - ❖ Find the training example \mathbf{x}_i that is *closest* to \mathbf{x}
 - ❖ Predict the label of \mathbf{x} to the label y_i associated with \mathbf{x}_i

Example:
How would you color the blank circles?



K-Nearest Neighbors

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to define distance?

Distance between instances

- ❖ How do we measure distances between instances in vector space?
- ❖ In general, a good place to inject knowledge about the domain
- ❖ Behavior of this approach can depend on this

Distance between instances

Numeric features, represented as n dimensional vectors

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- ❖ Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$



Euclidean distance

Manhattan distance

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- ❖ Manhattan distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- ❖ L_p -norm

- ❖ Euclidean = L_2

- ❖ Manhattan = L_1

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

$P > 0$

Distance between instances

What about symbolic/categorical features?

Distance between instances

Symbolic/categorical features

Most common distance is the *Hamming distance*

- ❖ Number of bits that are different
- ❖ Or: Number of features that have a different value
- ❖ Example:

\mathbf{X}_1 : {Shape=Triangle, Color=Red, Location=Left, Orientation=Up}

\mathbf{X}_2 : {Shape=Triangle, Color=Blue, Location=Left, Orientation=Down}

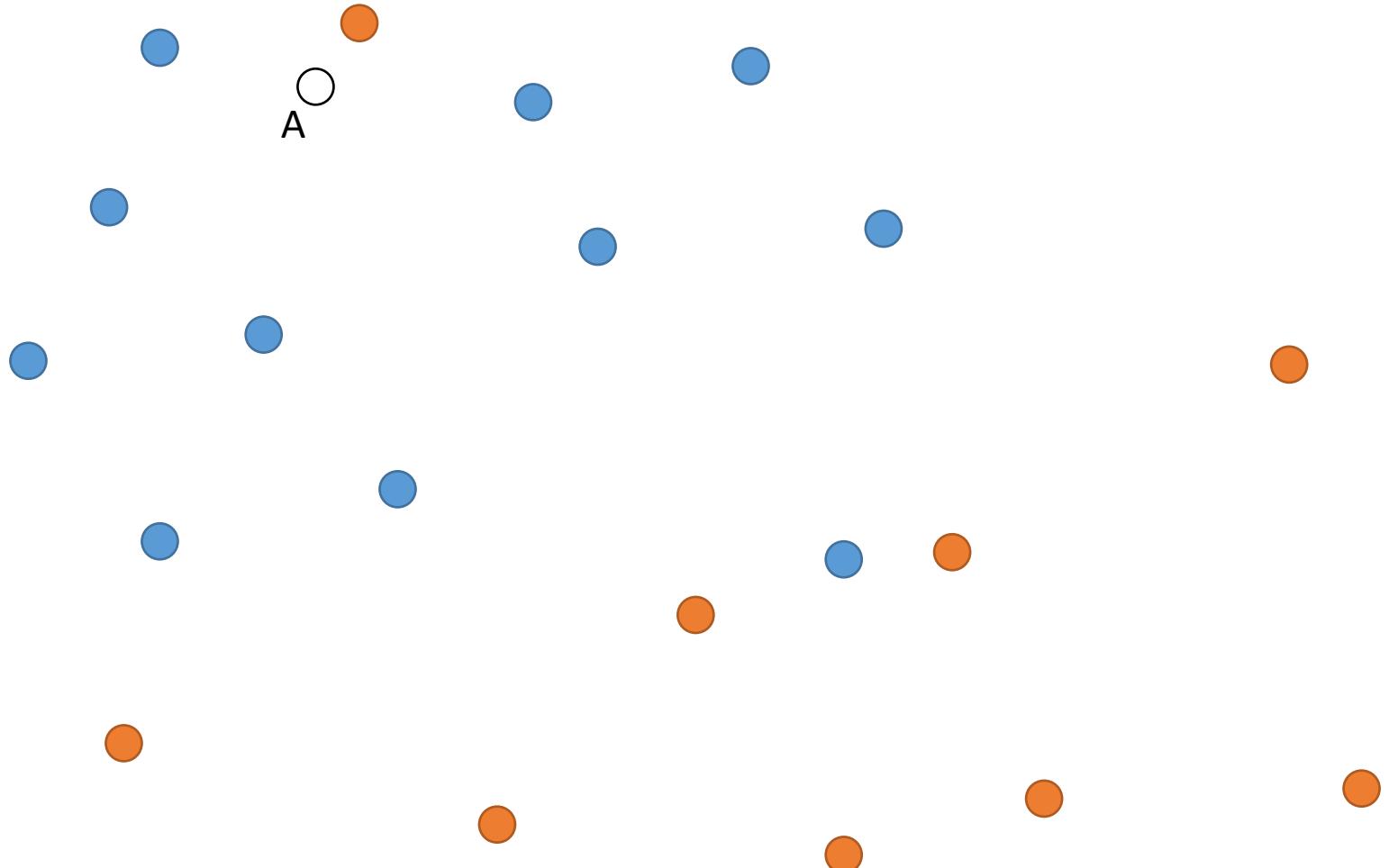
Hamming distance = 2

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to choose k and the distance measure?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Parameter K

What if K is too small?
What if K is too large?



Hyper-parameters in KNN

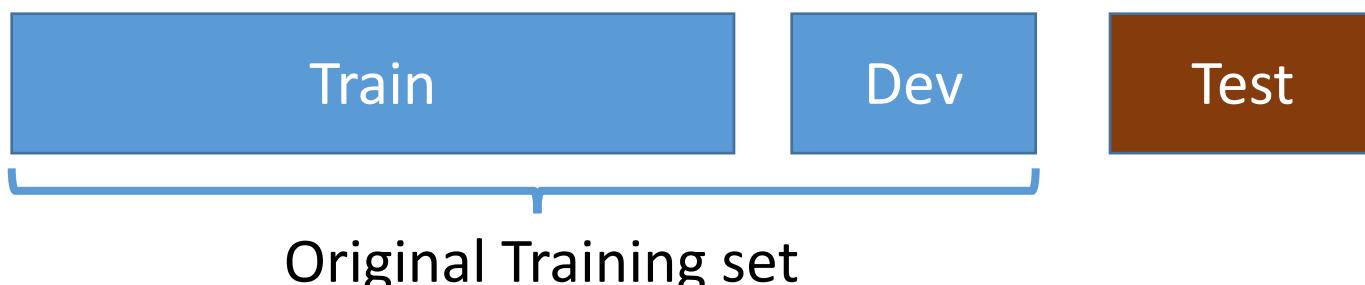
- ❖ (Hyper-)Parameters:
 - ❖ Choosing K (# nearest neighbors)
 - ❖ Distance measurement (e.g., p in the L_p -norm)

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

- ❖ Those are not specified by the algorithm itself
 - ❖ Require **empirical** studies
 - ❖ The best parameter set is **task/dataset-specific**.

Train/Dev/Test splits

- ❖ You are not allowed to look at Text in parameter tuning. Why?
- ❖ Split your training data into two sets:
 - ❖ Train: Training data (often 80-90%)
 - ❖ Dev: Development data (10-20%)
- ❖ Use Dev set (a.k.a. validation set) to find the best parameters



Recipe of train/dev/test

- ❖ For each possible value of the hyper-parameter (e.g., $M = 1, 2, 3, \dots, 10$)
 - ❖ Train a model using D^{TRAIN}
 - ❖ Evaluate the performance on D^{DEV}
- ❖ Choose the model parameter with the best performance on D^{DEV}
- ❖ (optional) Re-train the model on $D^{TRAIN} \cup D^{DEV}$ with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Tradeoff between Train v.s. Dev Size

- ❖ Consider having 120 data points; 20 data points are reserved for testing
 - ❖ What is the best way to split the remainder?
 - ❖ (A) # instances: Train: 95 Dev: 5 ?
 - ❖ (B) # instances: Train: 60 Dev: 40 ?

Trade off in Train/Dev splits

- ❖ Large Train, small Dev
(e.g., #train = 95, #dev = 5)



Result on dev is not represented

- ❖ Small Train, Large Dev
(e.g., #train = 60, #dev = 40)



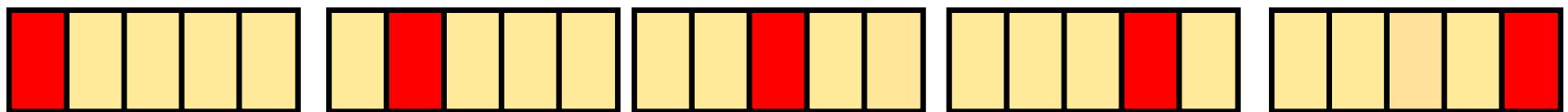
No enough data to train a model

N-fold cross validation

- ❖ Instead of a single training-dev split:



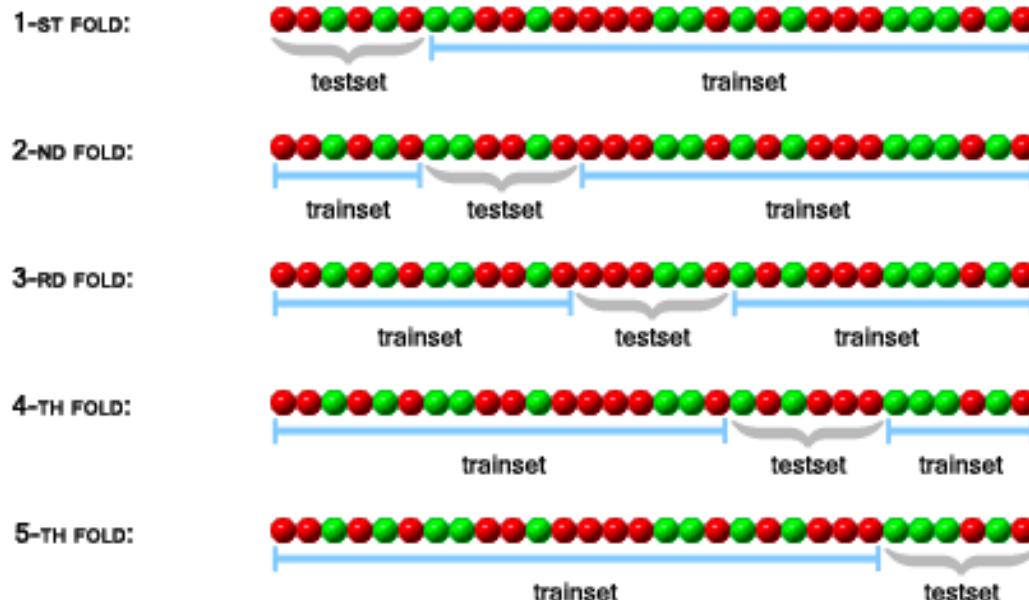
- ❖ Split data into N equal-sized parts



- ❖ Train and test N different classifiers
- ❖ Report average accuracy and standard deviation of the accuracy

Example

ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:



<https://genome.tugraz.at/proclassify/help/pages/XV.html>

Parameter 1

Accuracy: 100%

Accuracy: 50%

Accuracy: 50%

Accuracy: 100%

Accuracy: 100%

Parameter 2

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 50%

Avg Accuracy: 80%

Avg Accuracy: 90%

Finding parameters based on cross validation

- ❖ Given D^{TRAIN} and D^{TEST} , for each possible value of the hyper-parameter (e.g., $K = 1, 2, 3, \dots, 10$)
 - ❖ Conduct cross validation on D^{TRAIN} with parameter K
- ❖ Choose the model parameter with the best cross validation performance
- ❖ (Optional) Re-train the model on D^{TRAIN} with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to aggregate the information and make the prediction?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Q: other alternatives?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Neighbors' labels could be weighted by their distance
Related to Kernel method

Issues in designing KNN algorithm (computation/algorithim)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to store data?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to find the closest points?

Issues in designing KNN algorithm (computation/algorithms)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to store data?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the

How to find the closest points?

This is an important research topic, but I will not cover it in this class.
Reference: e.g. K-d tree (https://en.wikipedia.org/wiki/K-d_tree)

K-Nearest Neighbor

Fall 2022

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Learning Objectives

- ❖ KNN Algorithms
- ❖ Hyper-parameter tuning
 - ❖ Train/dev/test
 - ❖ N-fold cross-validation
- ❖ Decision boundary
- ❖ Curse of dimensionality
- ❖ Practical concerns -- Data preprocessing

KNN algorithm

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Inductive bias of KNN

Definition of inductive bias:

The set of assumptions that the learner uses to predict outputs of unseen inputs

- ❖ Label of point (data instance) is similar to the label of nearby points.

Example: Recognizing flowers

Types of Iris: **setosa, versicolor, and virginica**



(A)



(B)

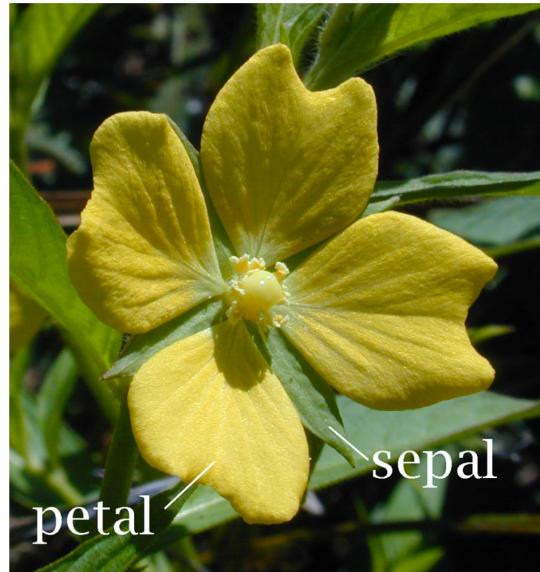


(C)



Iris dataset

- ❖ Features: the widths and lengths of sepal and petal



Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188

Understand dataset

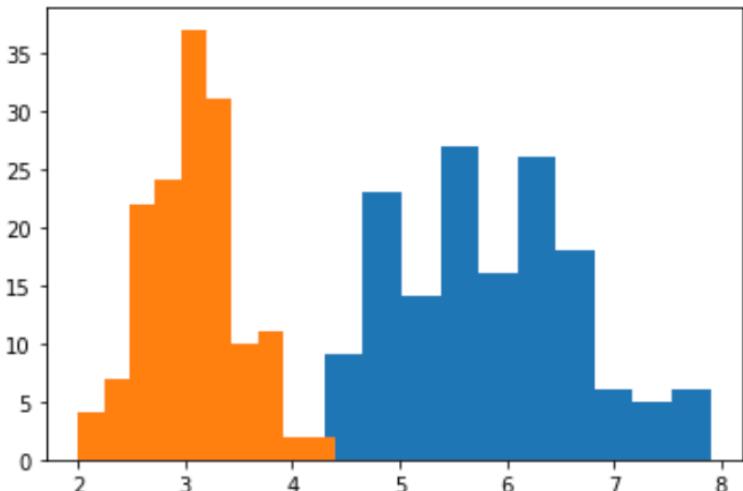
- ❖ <https://bit.ly/CM146-KNN-IRIS>

Fisher's *Iris* Data

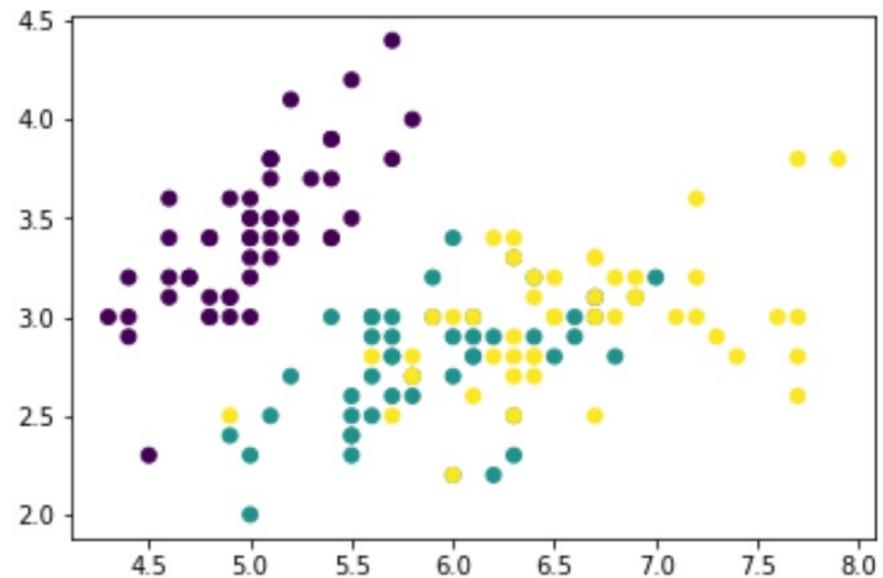
Sepal length ↴	Sepal width ↴	Petal length ↴	Petal width ↴	Species ↴
5.1	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
5.0	3.6	1.4	0.2	<i>I. setosa</i>
5.4	3.9	1.7	0.4	<i>I. setosa</i>
4.6	3.4	1.4	0.3	<i>I. setosa</i>

```
import matplotlib.pyplot as plt
import sklearn
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

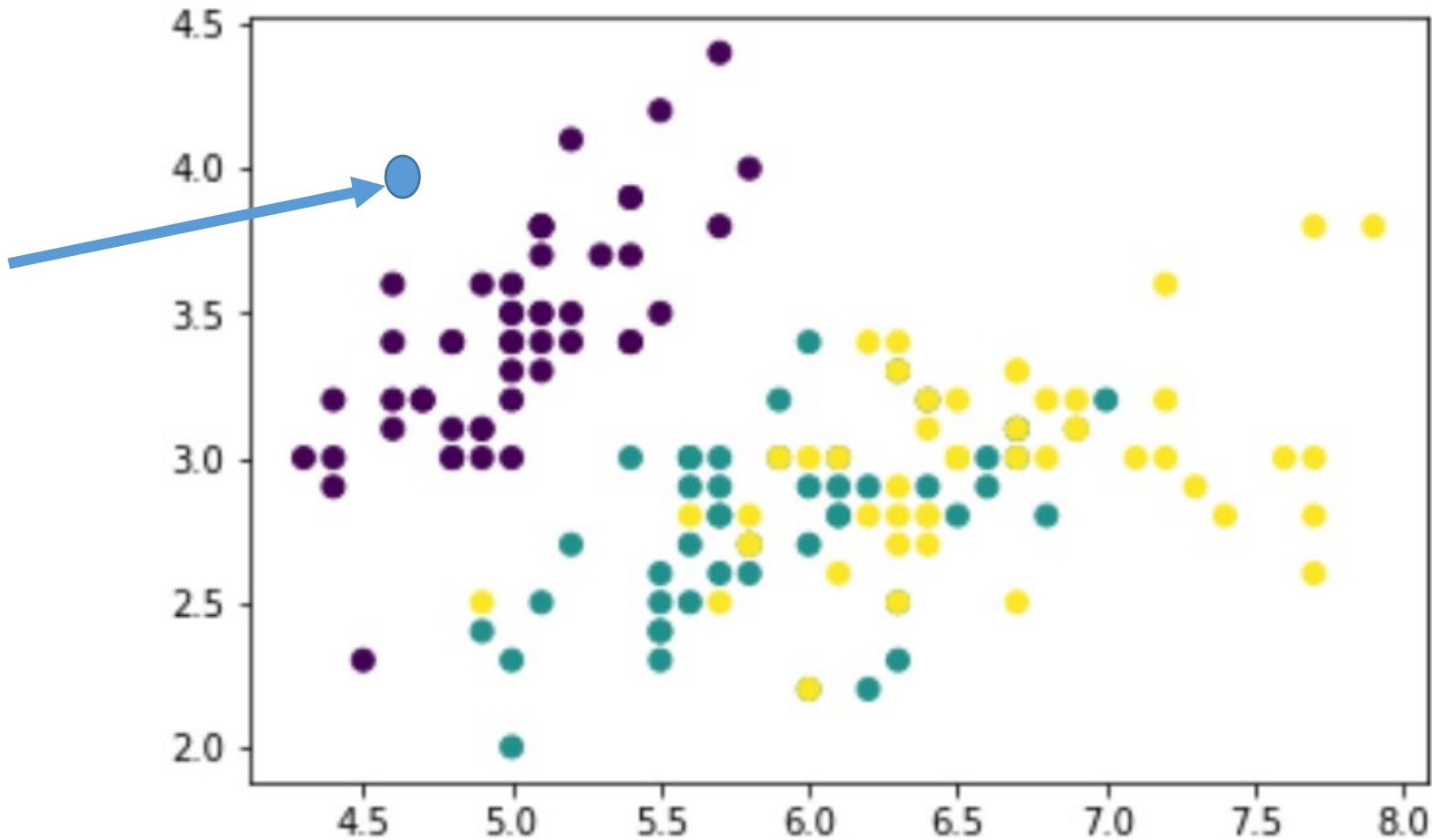
```
plt.figure()
plt.hist(X[:, 0]);
plt.hist(X[:, 1]);
```



```
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



```
plt.figure()  
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



Example: using 2 features

- ❖ Training data

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- ❖ Test data

petal width = 3 and sepal width = 6

- ❖ Let's use L1 (Manhattan) distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

Example: using 2 features

❖ Training data

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

❖ Test data

petal width = 3 and sepal width = 6

Category (y)	L1 Distance
setosa	$1+1=2$
versicolor	$2+0=2$
virginica	$0+1=1$



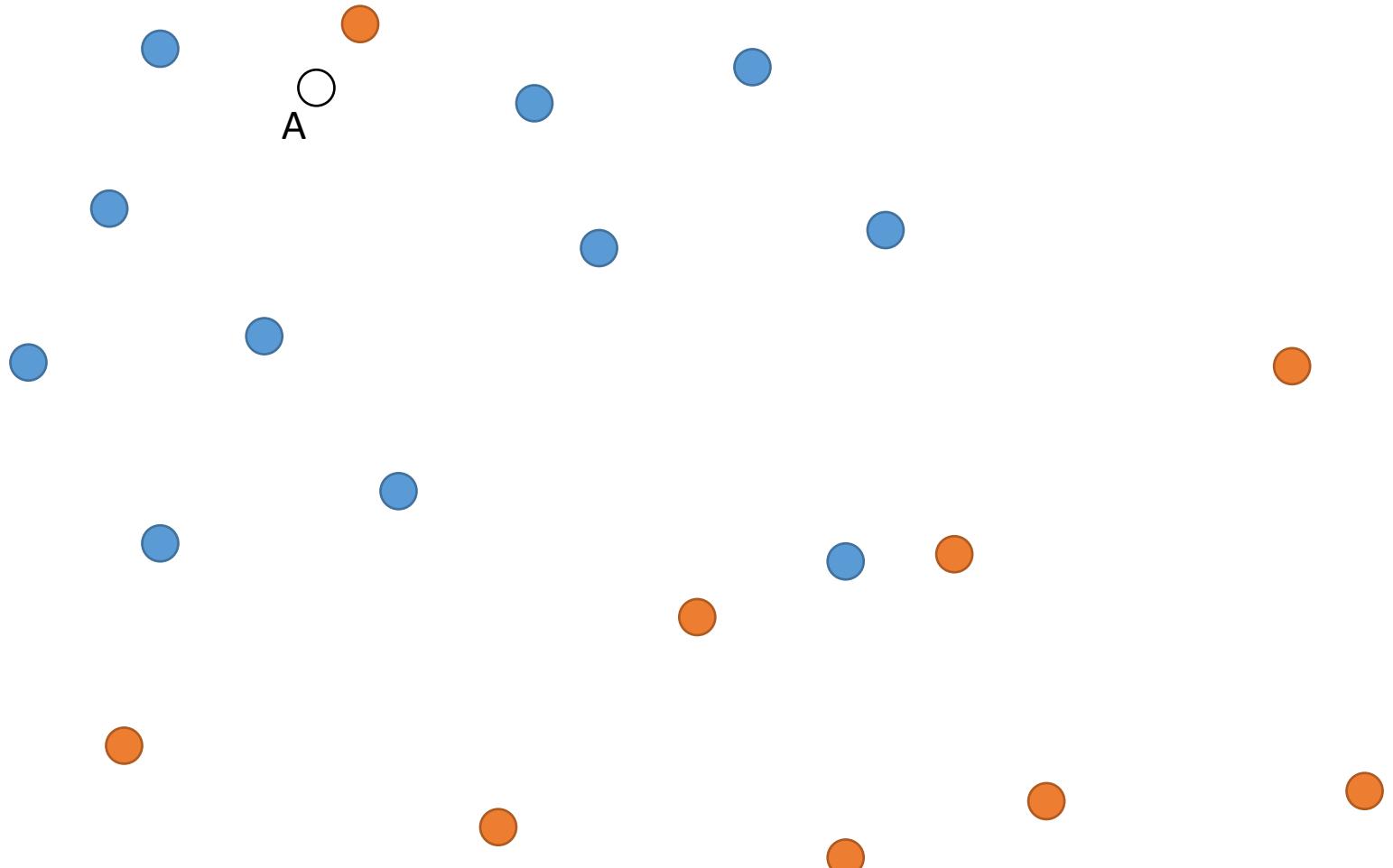
Hyper-Parameters & Design Choices

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
- ❖ Learning: Just store all the training examples
 - How to choose k and the distance measure?
- ❖ Prediction
 - new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Hyper-Parameter K

What if K is too small?
What if K is too large?



Hyper-parameters in KNN

- ❖ Hyper-parameters:
 - ❖ Choosing K (# nearest neighbors)
 - ❖ Distance measurement (e.g., p in the L_p -norm)
$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$
- ❖ Those are not specified by the algorithm itself
 - ❖ Require **empirical** studies
 - ❖ The best parameter set is **task/dataset-specific**.

Train/Dev/Test splits

- ❖ Split your data into Train/Dev/Test
 - ❖ Only use Train and Dev for developing models. Why?
- ❖ Train: Data for training models
- ❖ Dev (a.k.a. validation set): find the best parameters by evaluating models on dev
- ❖ Test: report the performance

Practice exam

Final exam



Labeled data

Recipe of train/dev/test

- ❖ For each possible value of the hyper-parameter (e.g., $M = 1, 2, 3, \dots, 10$)
 - ❖ Train a model using D^{TRAIN}
 - ❖ Evaluate the performance on D^{DEV}
- ❖ Choose the model with the best performance on D^{DEV}
- ❖ Evaluate the final model on D^{TEST}

Recipe of train/dev/test

- ❖ For each possible value of the hyper-parameter (e.g., $M = 1, 2, 3, \dots, 10$)
 - ❖ Train a model using D^{TRAIN}
 - ❖ Evaluate the performance on D^{DEV}
- ❖ Choose the model with the best performance on D^{DEV}
- ❖ (optional) Re-train the model on $D^{TRAIN} \cup D^{DEV}$ with the best hyper-parameter set
- ❖ Evaluate the final model on D^{TEST}

Tradeoff between Train v.s. Dev Size

- ❖ Consider having 120 data points; 20 data points are reserved for testing
 - ❖ What is the best way to split the remainder?
 - ❖ (A) # instances: Train: 95 Dev: 5 ?
 - ❖ (B) # instances: Train: 60 Dev: 40 ?

Trade off in Train/Dev splits

- ❖ Large Train, small Dev
(e.g., #train = 95, #dev = 5)



Result on dev is not representative

- ❖ Small Train, Large Dev
(e.g., #train = 60, #dev = 40)



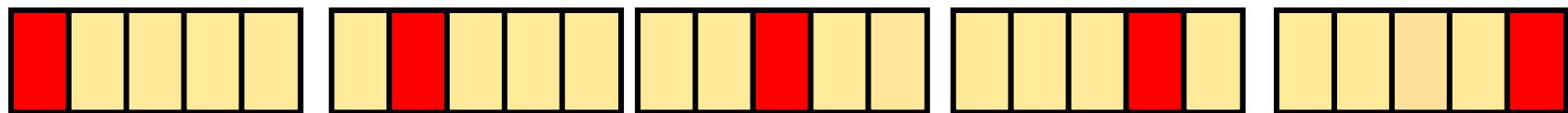
No enough data to train a model

N-fold cross validation

- ❖ Instead of a single training-dev split:



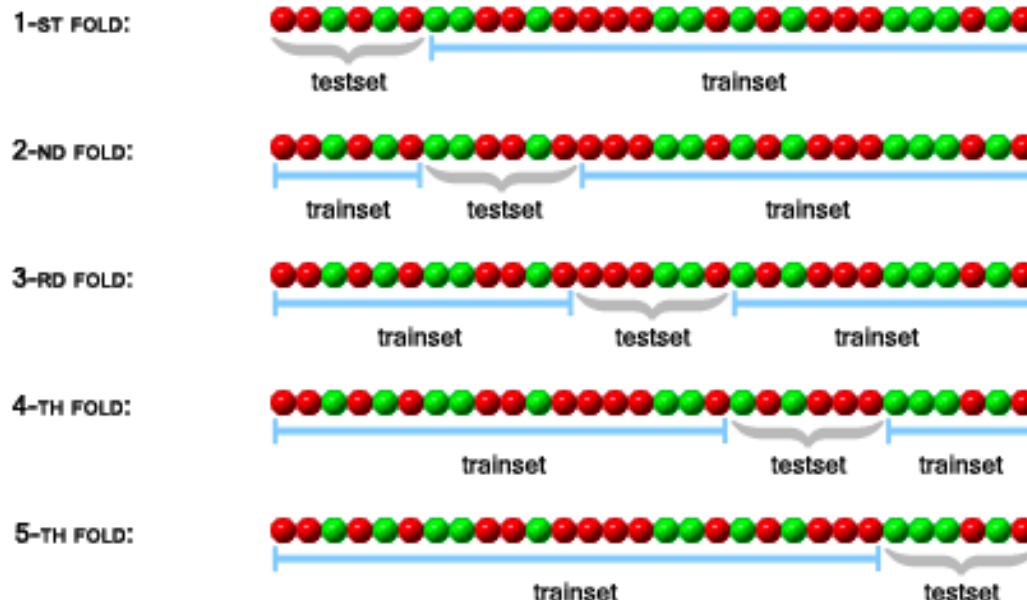
- ❖ Split data into N equal-sized parts



- ❖ Train and test N different classifiers
- ❖ Report average accuracy and standard deviation of the accuracy

Example

ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:



<https://genome.tugraz.at/proclassify/help/pages/XV.html>

Parameter 1

Accuracy: 100%

Accuracy: 50%

Accuracy: 50%

Accuracy: 100%

Accuracy: 100%

Parameter 2

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 50%

Avg Accuracy: 80%

Avg Accuracy: 90%

Finding parameters based on cross validation

- ❖ Given D^{TRAIN} and D^{TEST} , for each possible value of the hyper-parameter (e.g., $K = 1, 2, 3, \dots, 10$)
 - ❖ Conduct cross validation on D^{TRAIN} with parameter K
- ❖ Choose the model parameter with the best cross validation performance
- ❖ (Optional) Re-train the model on D^{TRAIN} with the best parameter set
- ❖ Evaluate the model on D^{TEST}

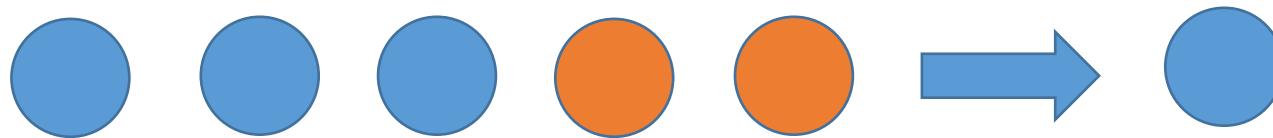
Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to aggregate the information and make the prediction?

Construct the label of x using these k points

- ❖ Majority vote



To break ties, it's better to use odd K

- ❖ Weighted vote

- ❖ Weight by their distances; this is related to kernel methods (will talk about this later)

Issues in designing KNN algorithm (computation/algorithim)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to store data?

How to find the closest points?

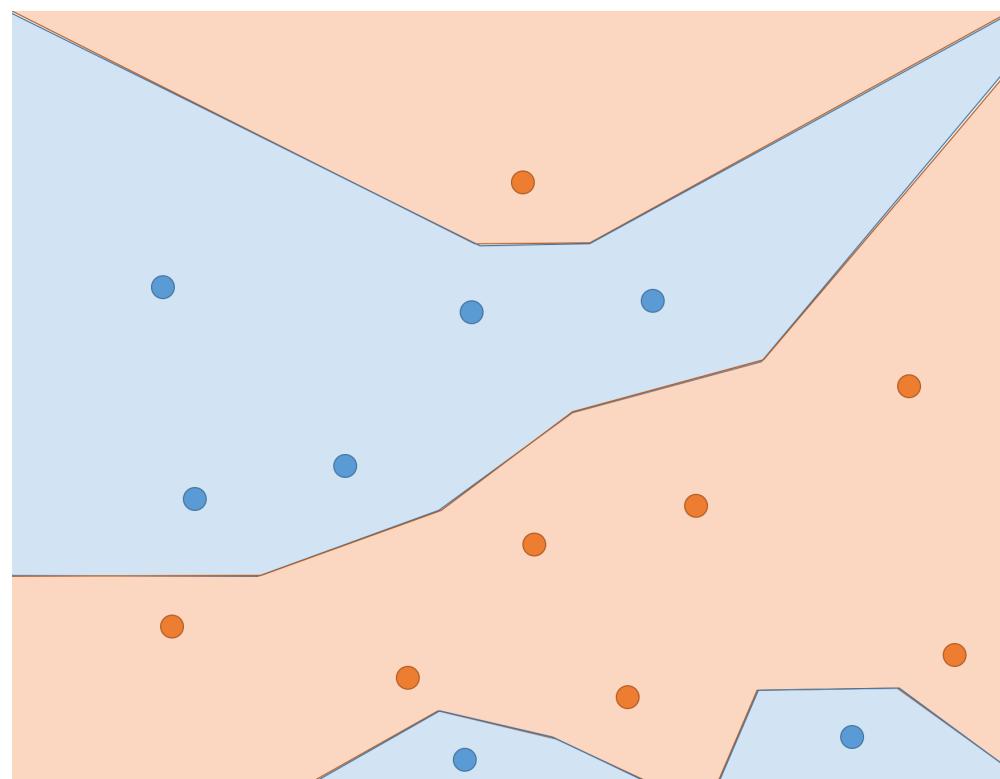
Issues in designing KNN algorithm (computation/algorithim)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to store data?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the

How to find the closest points?

This is an important research topic, but I will not cover it in this class.
Reference: e.g. K-d tree (https://en.wikipedia.org/wiki/K-d_tree)

Decision Boundary



The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

- ❖ **No**, it never forms an explicit hypothesis

Given a training set what is the implicit function that is being computed

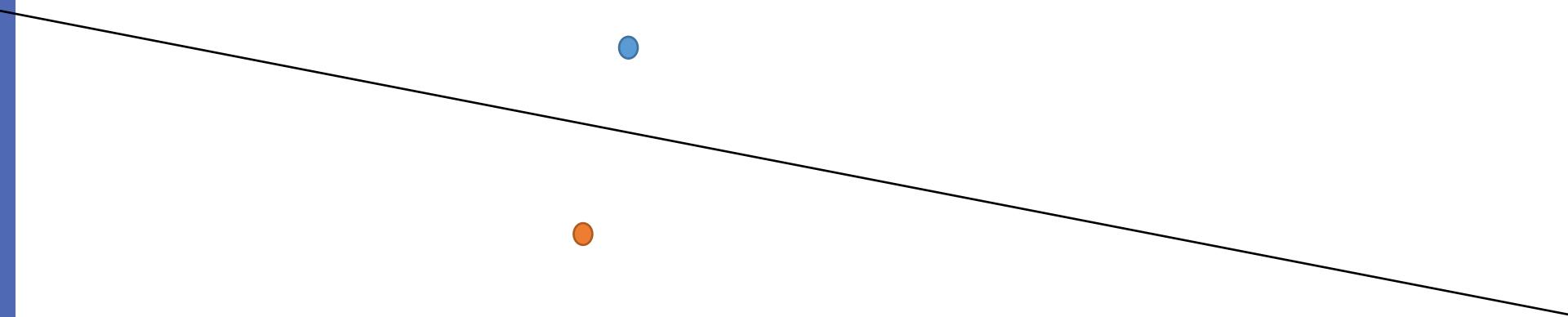
Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



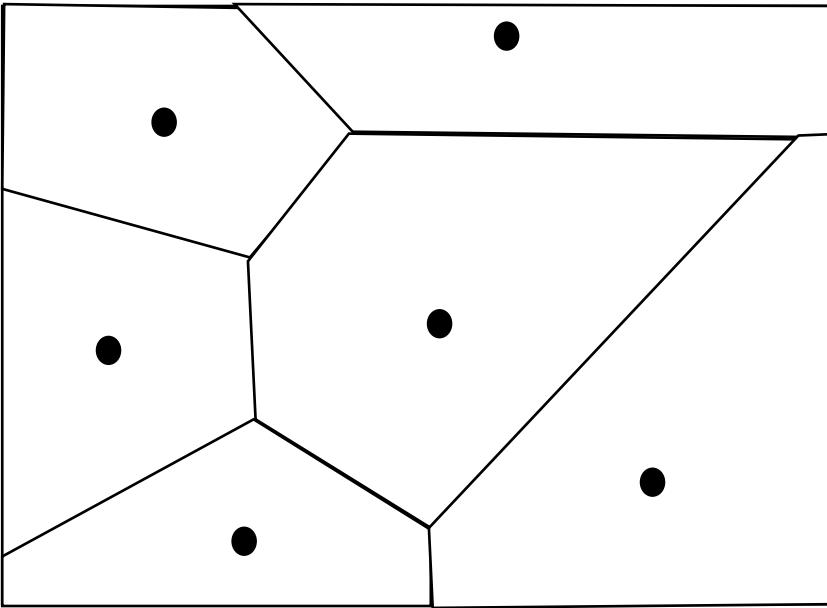
Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



- ❖ A line bisecting the two points

The Voronoi Diagram (w/ Euclidean distance)

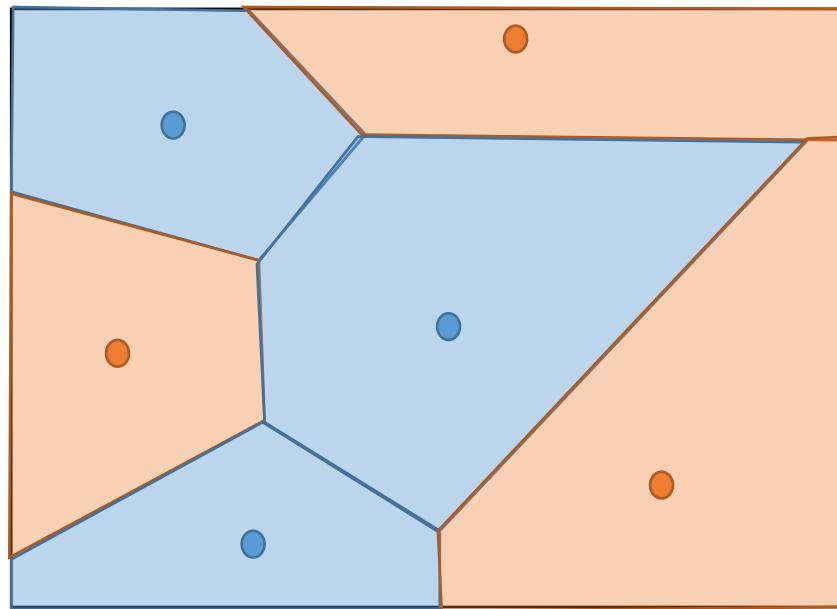


For any point x in a training set S ,
the **Voronoi Cell** of x is a polytope consisting of all points closer to x than any other points in S

The **Voronoi diagram** is the union of all Voronoi cells

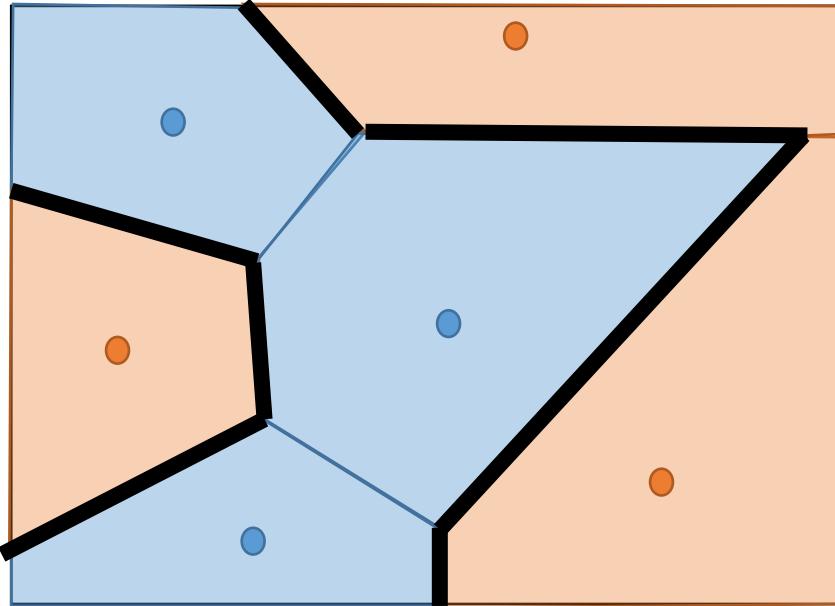
- Covers the entire space

Voronoi diagrams of training examples



Voronoi diagrams colored with the output label
Picture uses Euclidean distance with 1-nearest neighbor.

Decision boundary of 1-NN



What about K-nearest neighbors?

Also partitions the space, but much more complex
decision boundary

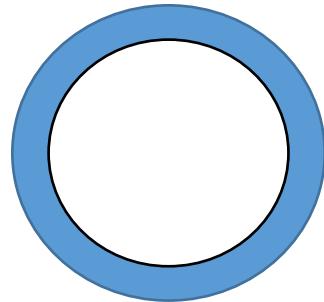


<https://erikbern.com/2015/10/20/nearest-neighbors-and-vector-models-epilogue-curse-of-dimensionality.html>

The Curse of Dimensionality

Example : What fraction of the volume of a unit sphere lies between radius $1 - \epsilon$ and radius 1?

In two dimensions



$$V = \pi R^2$$

volume radius

Arrows point from the words "volume" and "radius" to the corresponding terms in the equation $V = \pi R^2$.

What fraction of the area of the circle is in the blue region?

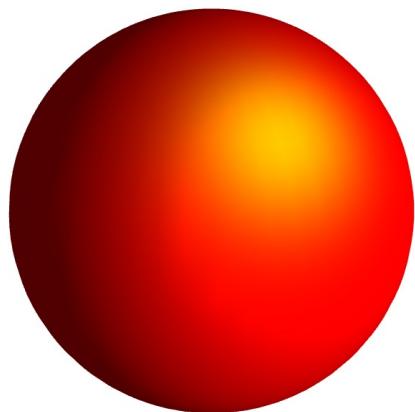
$$\frac{\pi \cdot 1^2 - \pi(1 - \epsilon)^2}{\pi \cdot 1^2} = 1 - (1 - \epsilon)^2$$

The Curse of Dimensionality

Example : What fraction of the volume of a unit sphere lies between radius $1 - \epsilon$ and radius 1? Type equation here.

In three dimensions

$$V = \frac{4\pi}{3} R^3$$



$$\frac{\frac{4\pi}{3} 1^3 - \frac{4\pi}{3} (1 - \epsilon)^3}{\frac{4\pi}{3} 1^3} = 1 - (1 - \epsilon)^3$$

The Curse of Dimensionality

Example : What fraction of the volume of a unit sphere lies between radius $1 - \epsilon$ and radius 1?

In two dimensions $\frac{\pi \cdot 1^2 - \pi(1 - \epsilon)^2}{\pi \cdot 1^2} = 1 - (1 - \epsilon)^2$

In d dimensions $1 - (1 - \epsilon)^d$

When d is large this fraction goes to 1!

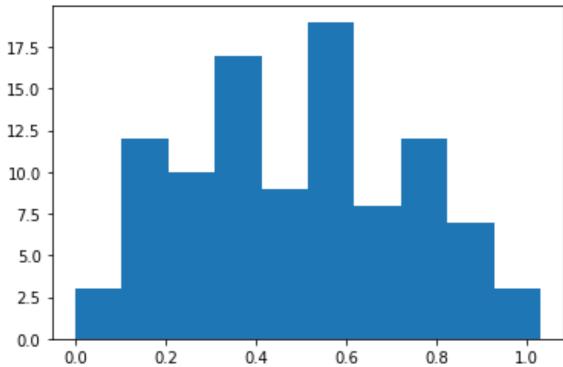
In high dimensions, most of the volume of the sphere is far away from the center!

Demo

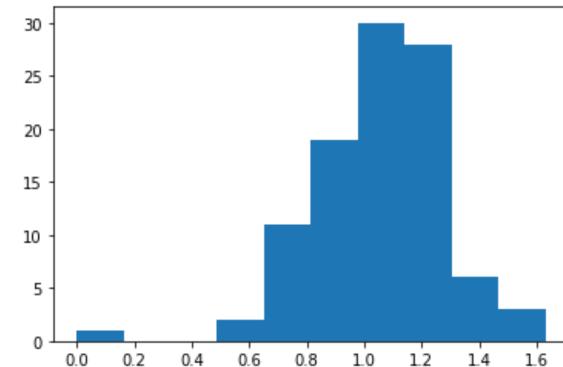
❖ <https://bit.ly/CM146-KNN-dim>

```
def fun(num, dim):
    X = np.random.rand(num, dim)
    a = [np.linalg.norm(X[0,:]-X[i,:]) for i in range(num)]
    plt.hist(a)
```

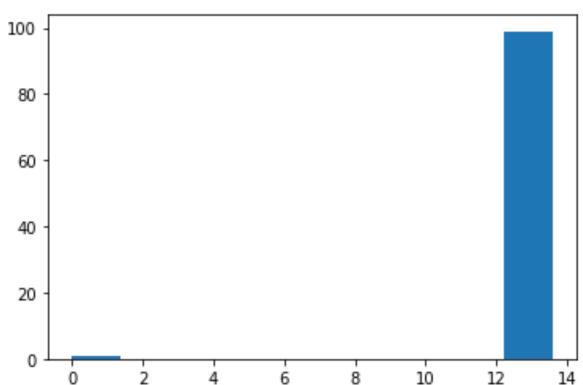
fun(100, 2)



fun(100, 10)



fun(100, 1000)



The Curse of Dimensionality

- ❖ Most of the points in high dimensional spaces are far away from the origin!
- ❖ Need more data to “fill up the space”
- ❖ Bad news for k-NN in high dimensional spaces
 - ❖ Even if most/all features are relevant, in high dimensional spaces, most points are equally far away from each other!

Dealing with the curse of dimensionality

- ❖ Most “*real-world*” data is not uniformly distributed in the high dimensional space
- ❖ Capturing the *underlying dimensionality* of the space -- dimensionality reduction

Data Preprocessing



Image from https://twitter.com/cat_cooking/status/858502449149218816
Lec 4: KNN & Decision Tree

Example: using 2 features

- ❖ Training data (length in cm)

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- ❖ Test data

petal width = 3 and sepal width = 6

- ❖ Let's use L1 (Manhattan) distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

Example: using 2 features

- ❖ Training data (length in cm)

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- ❖ Test data

petal width = 3 and sepal width = 6

Category (y)	L1 Distance
setosa	$1+1=2$
versicolor	$2+0=2$
virginica	$0+1=1$



Example: using 2 features

1cm=10mm

❖ Training data

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica

❖ Test data

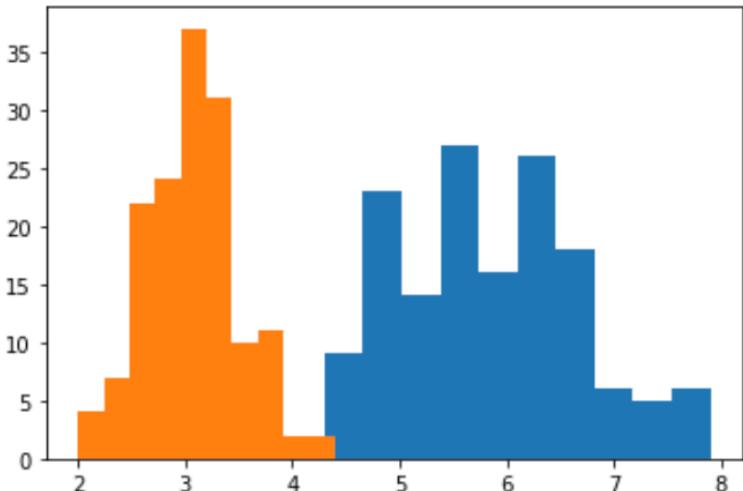
petal width = 3 and sepal width = 60

Category (y)	L1 Distance
setosa	$1+10=11$
versicolor	$2+0=2$
virginica	$0+10=10$

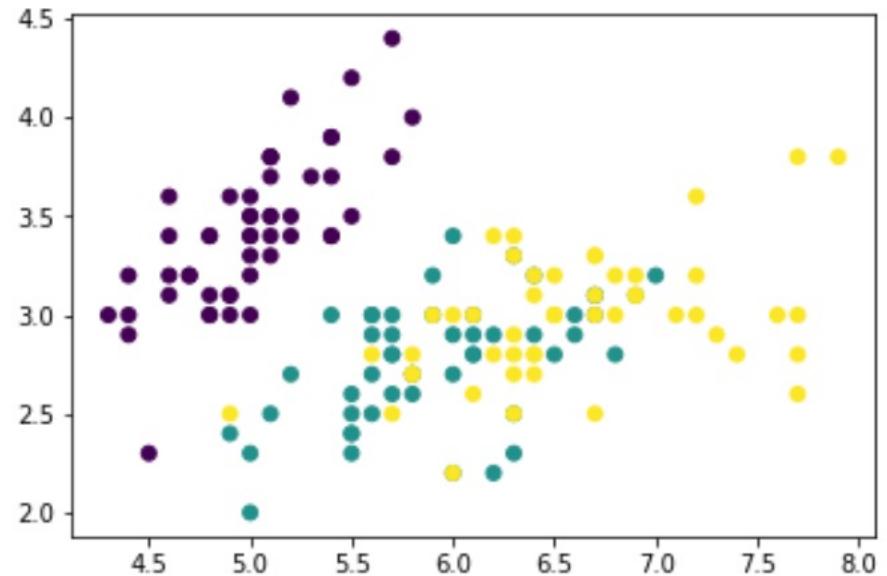


```
import matplotlib.pyplot as plt
import sklearn
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

```
plt.figure()
plt.hist(X[:, 0]);
plt.hist(X[:, 1]);
```



```
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



Preprocess data

- ❖ Normalize data to have zero mean and unit standard deviation in each dimension

$$\bar{x}_d = \frac{1}{N} \sum_n x_{nd}, \quad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- ❖ Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Example: using 2 features

❖ Training data

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica
Mean	2.67	53.3	
Std	1.53	5.77	

Example: using 2 features

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica
Mean	2.67	53.3	
Std	1.53	5.77	

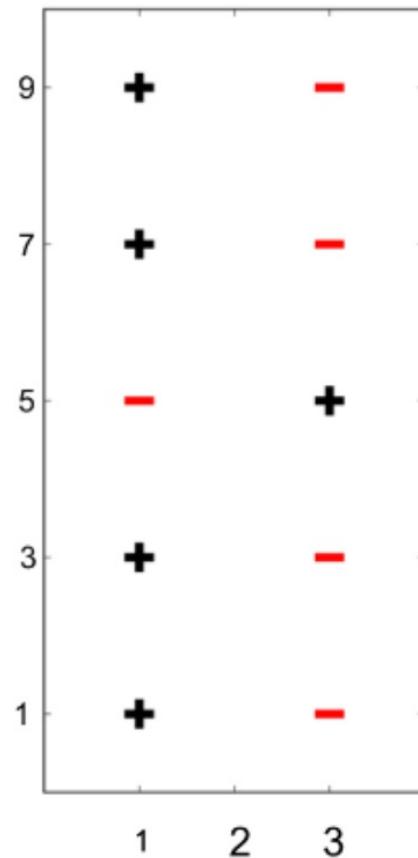
ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	0.87	-0.57	setosa
2	-1.09	1.15	versicolor
3	0.22	-0.57	virginica

Test data (after pre-processing):

petal width = 0.22 and sepal width = 1.15

Exercise

- 1) Draw the decision boundary of 1-NN
- 2) Draw the decision boundary of 3-NN



Decision Tree

Fall 2022

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

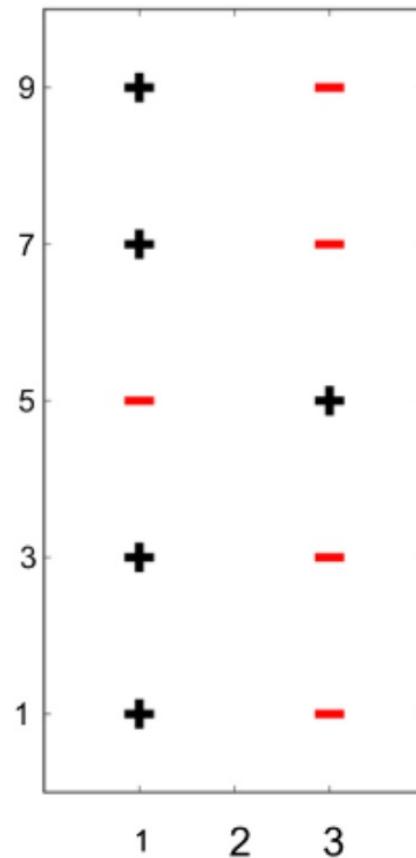
The instructor gratefully acknowledges Dan Roth, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

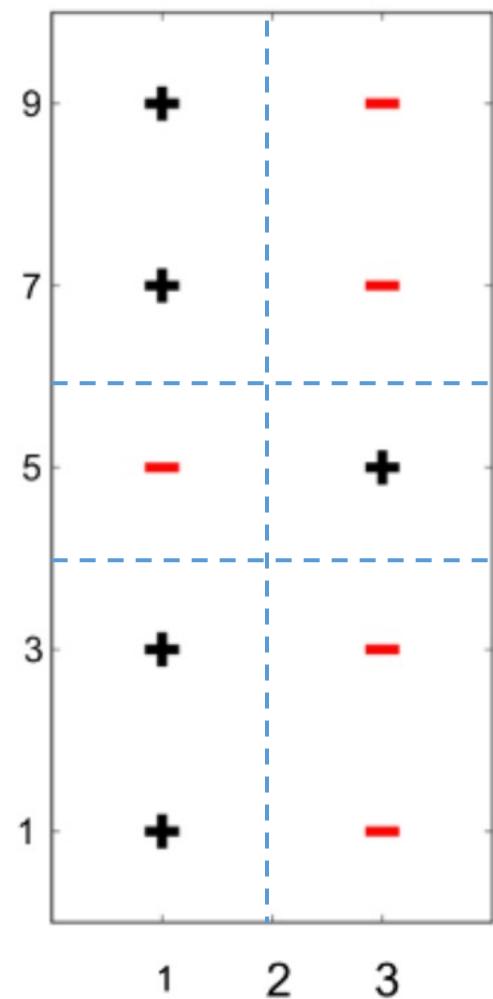
- ❖ Hw1: due 10/25 11:59pm PT
- ❖ Quiz1: due 10/11 (next Tue) 11:59pm PT
- ❖ PTEs

Exercise

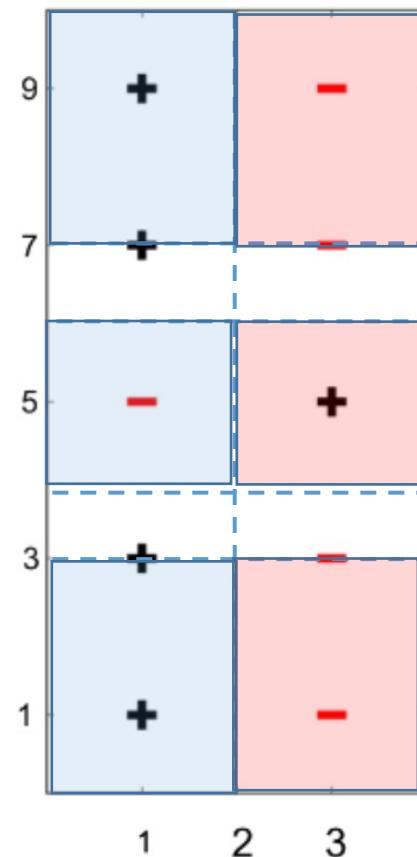
- 1) Draw the decision boundary of 1-NN
- 2) Draw the decision boundary of 3-NN



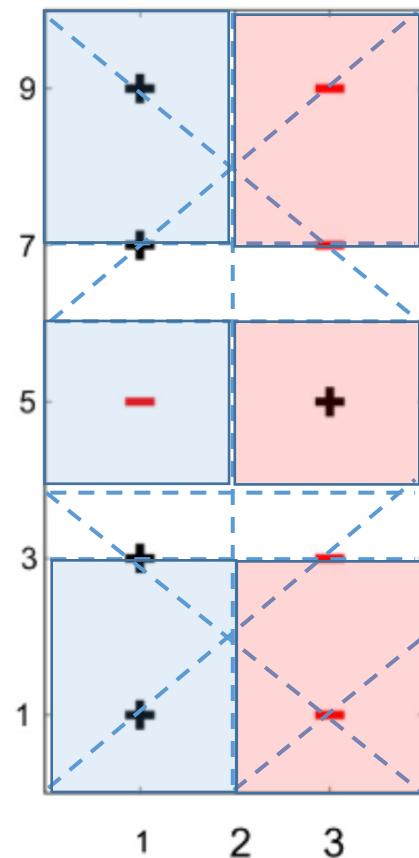
1-NN



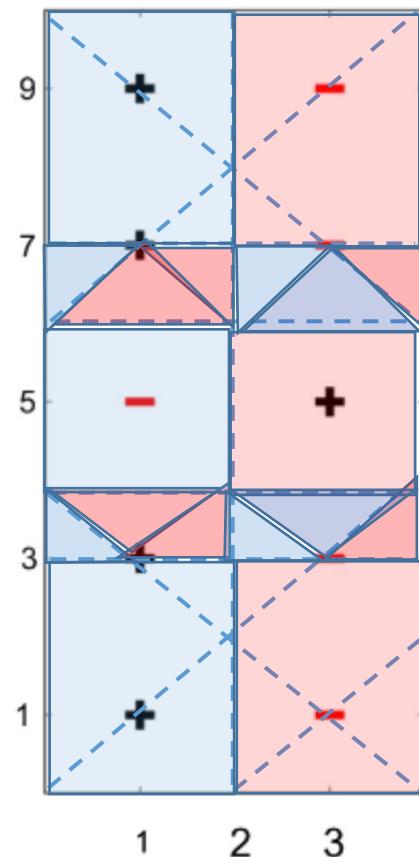
3-NN



3-NN



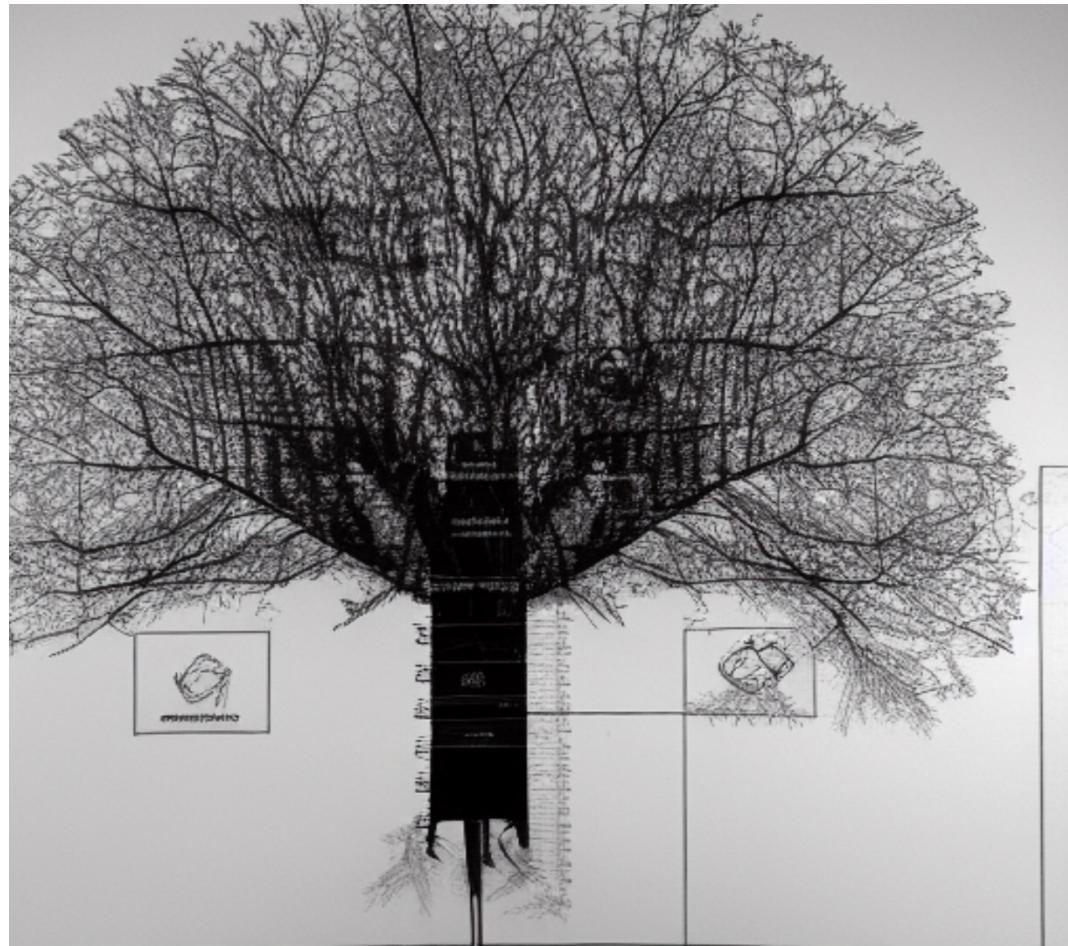
3-NN



This Lecture

- ❖ Model/Representation: Decision trees
- ❖ Algorithm: Learning decision trees
 - (ID3 algorithm)
 - ❖ Information theory / Entropy
 - ❖ Greedy heuristic (based on information gain)

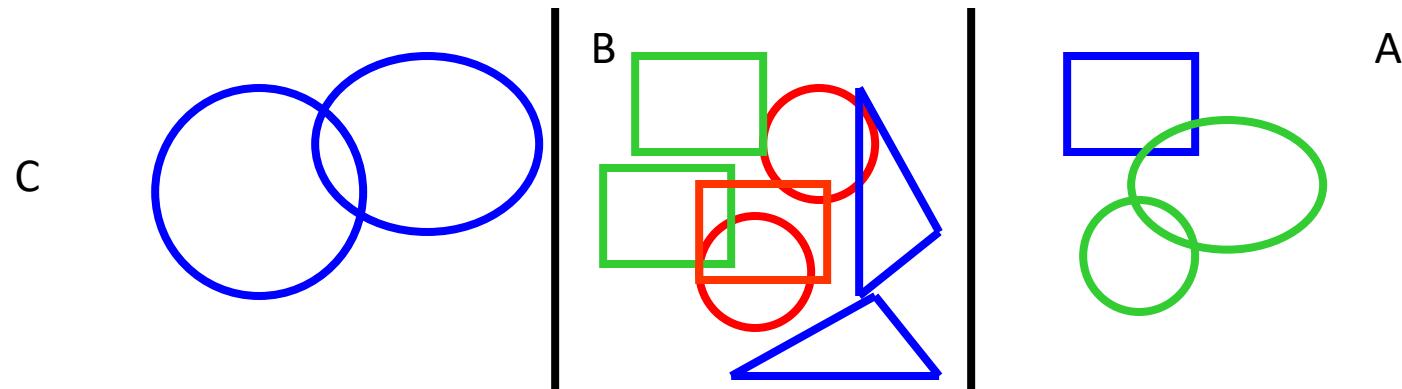
What is a decision tree?

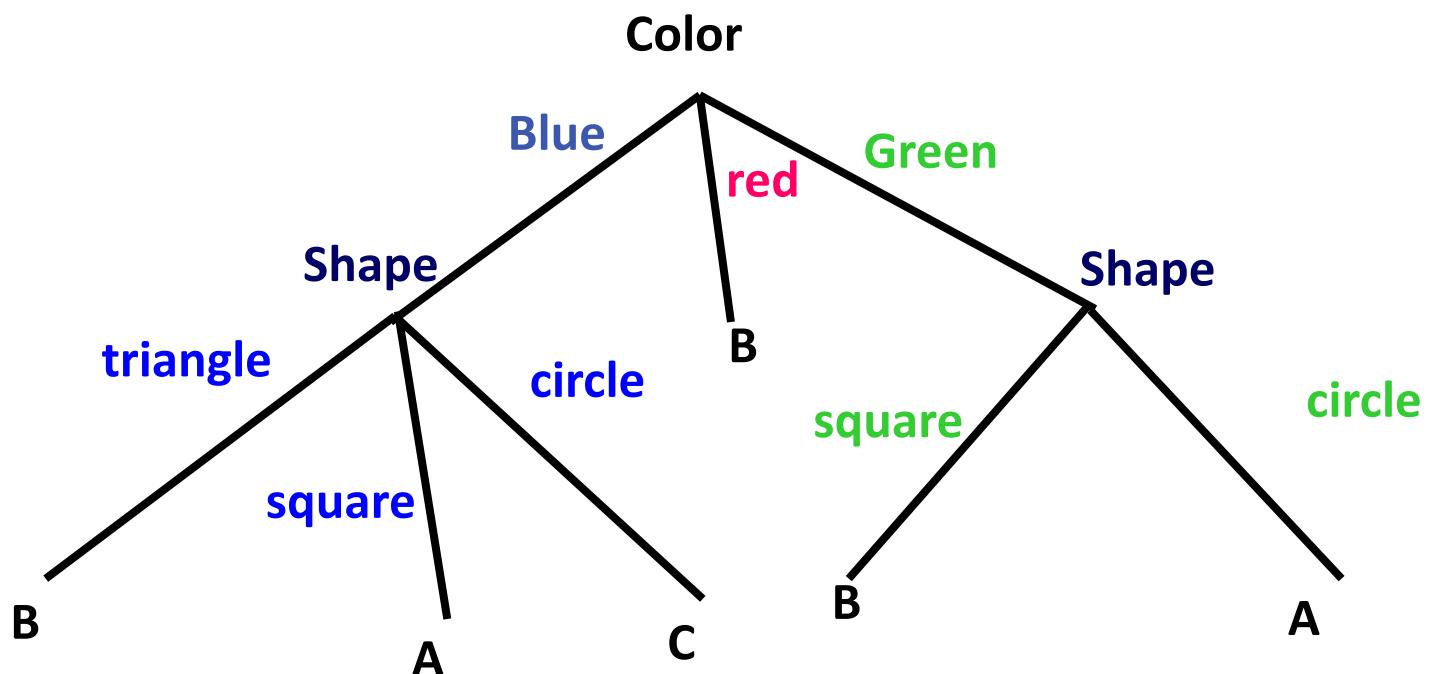
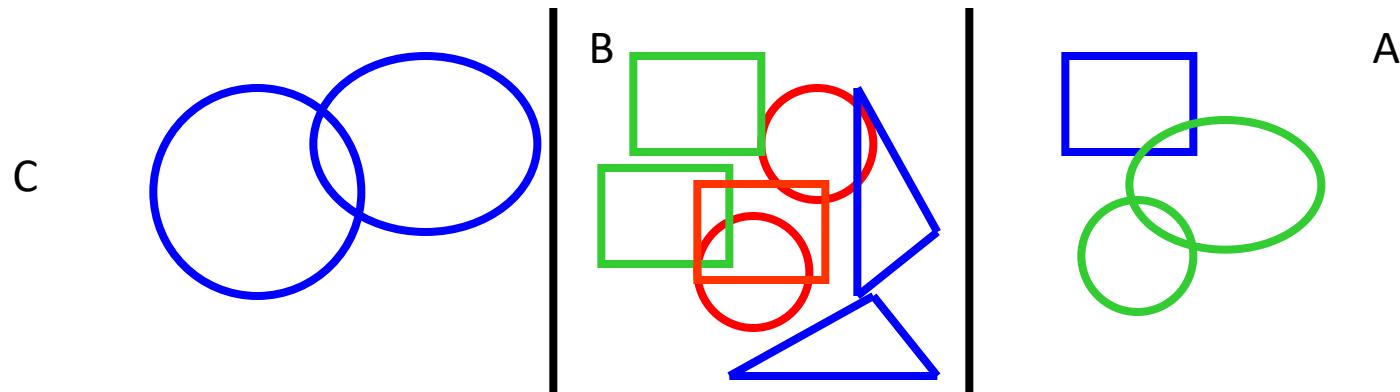


Generated by <https://beta.dreamstudio.ai/dream>

Sample dataset

- ❖ A hierarchical data structure that represents data
- ❖ What is the label for a red triangle?

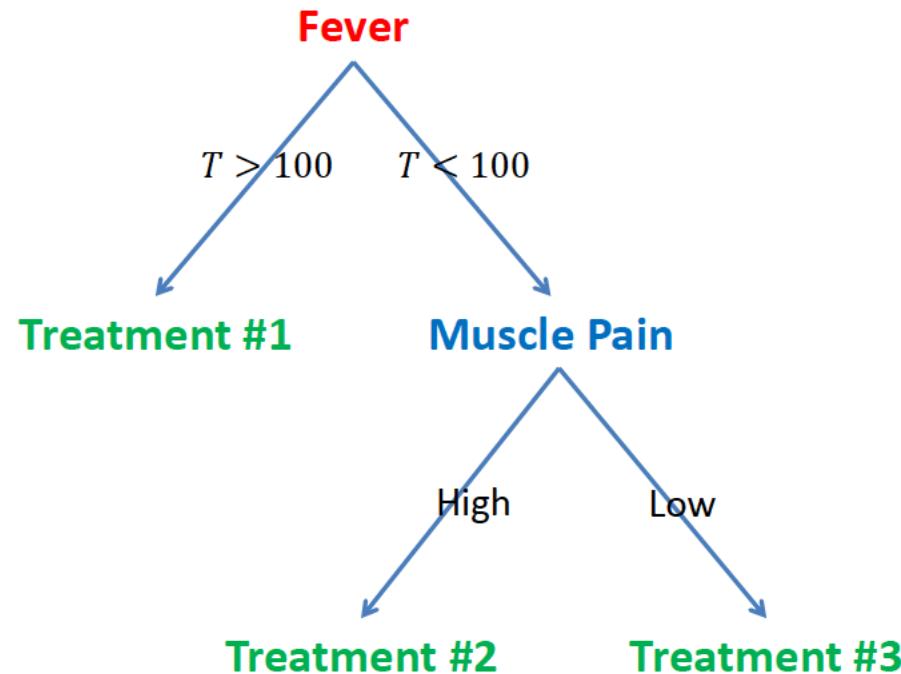




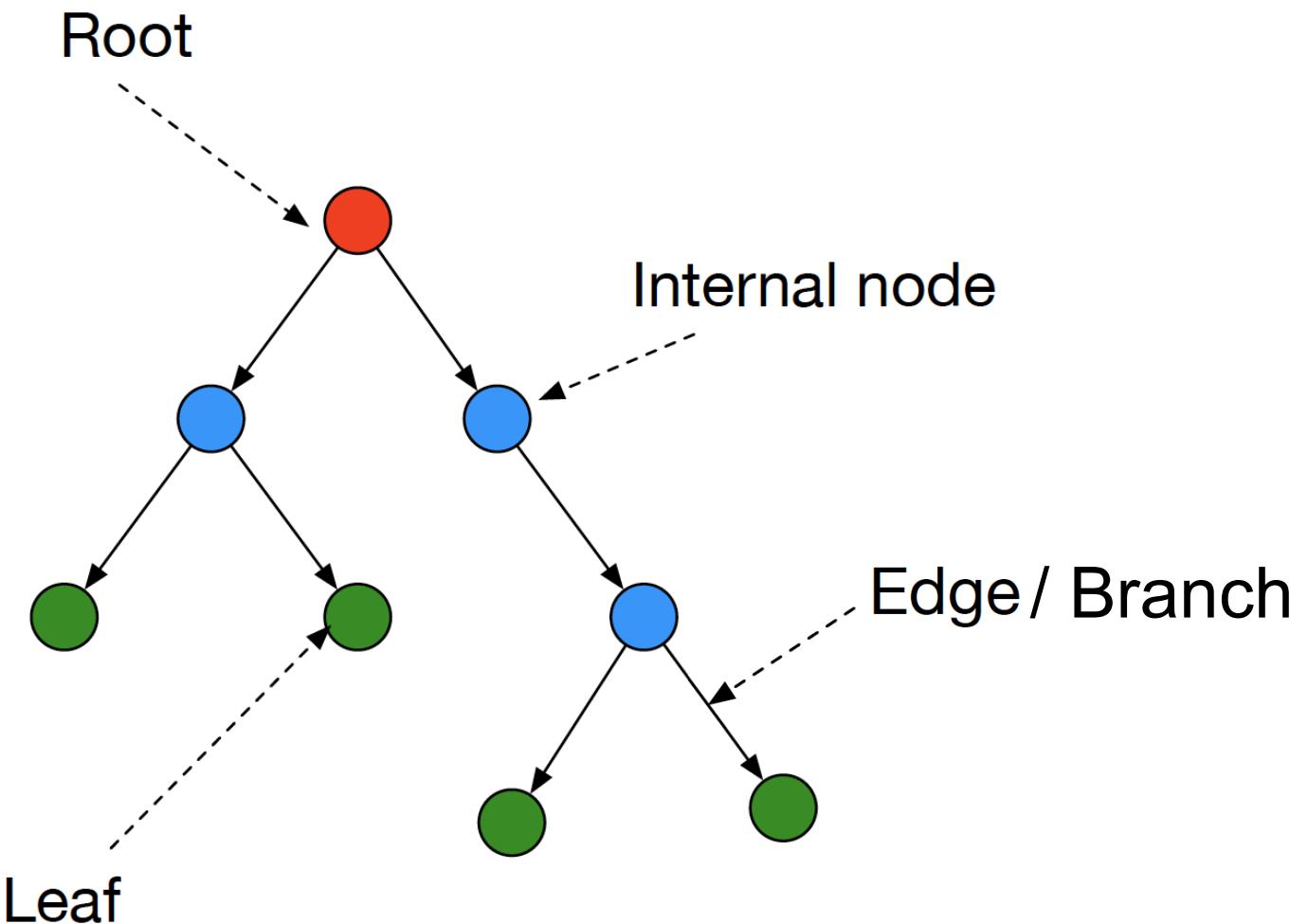
Motivations:

Many decisions are tree structures

Medical treatment



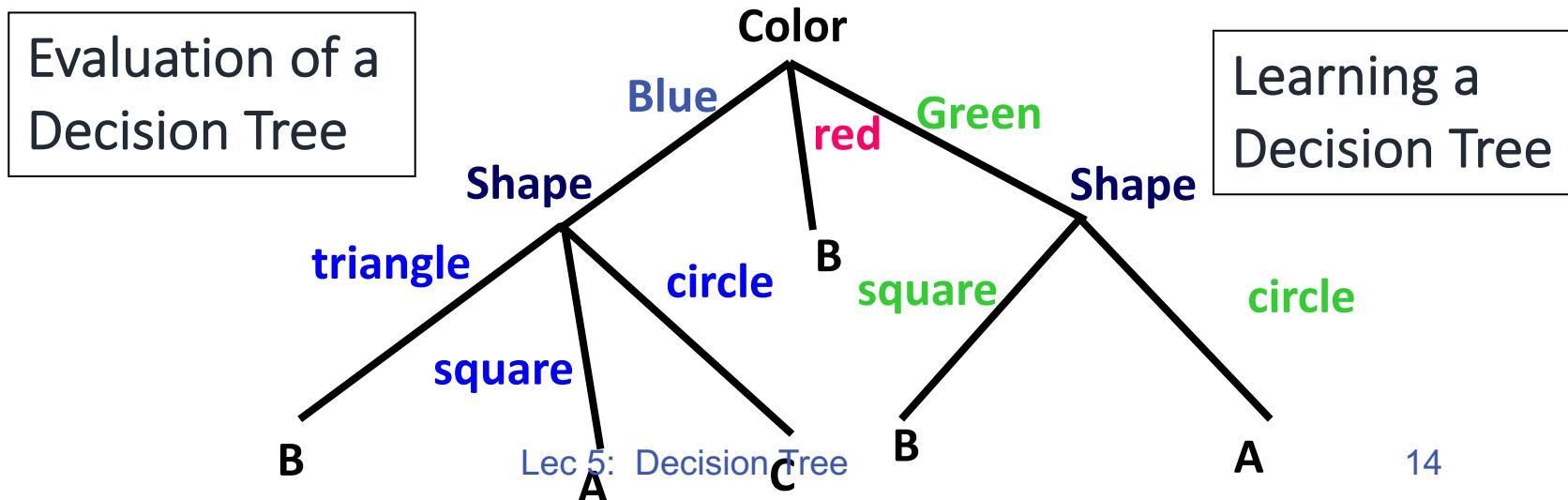
Terminology



Will sometimes drop the arrows on the edges

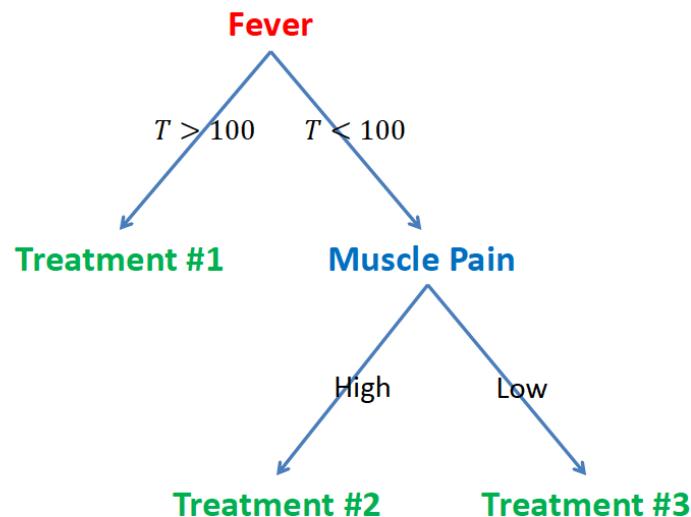
The Representation

- ❖ Decision Trees are classifiers for instances represented as feature vectors (color= ; shape= ; label=)
- ❖ Nodes are **tests** for feature values
- ❖ Edges: There is one branch for each value of the feature
- ❖ Leaves specify the category (labels)
- ❖ Can categorize instances into multiple disjoint categories

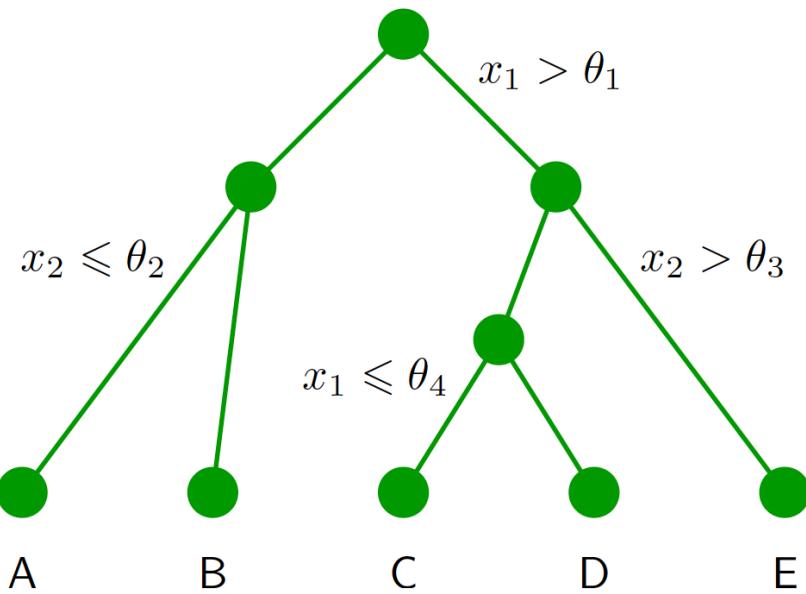
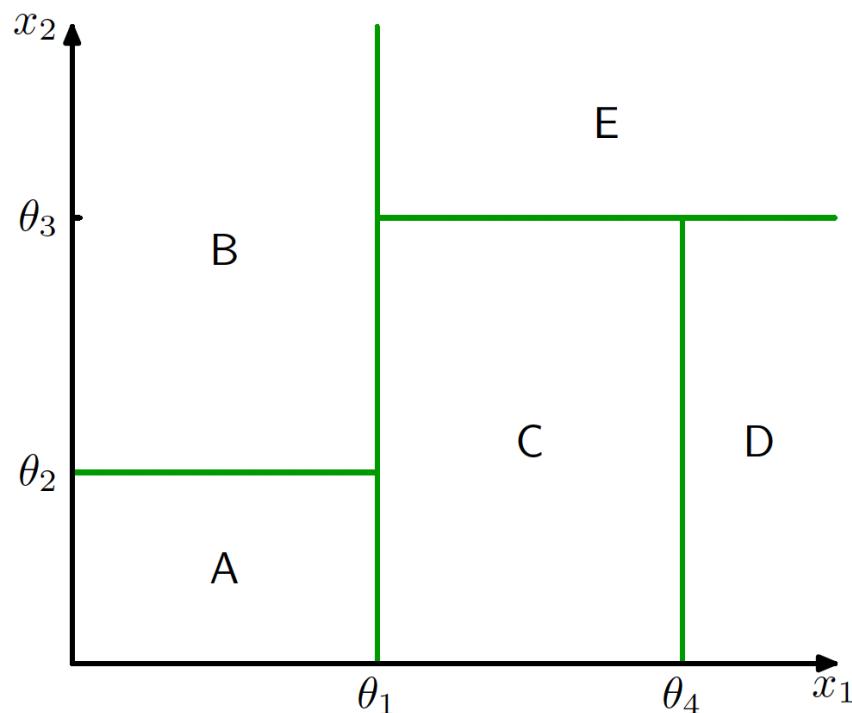


Handling real-valued features

- ❖ Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- ❖ Numerical values can be used by splitting nodes with thresholds

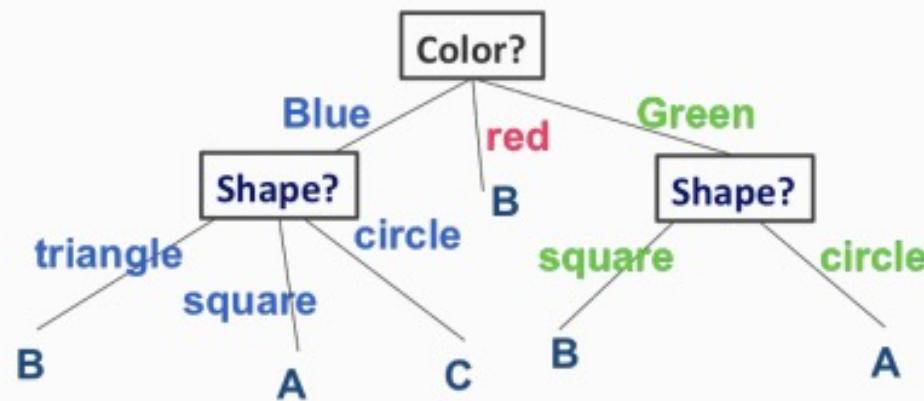


A tree partitions the feature space



Expressivity of Decision Trees

- ❖ What Boolean functions can decision trees represent?
 - any Boolean function

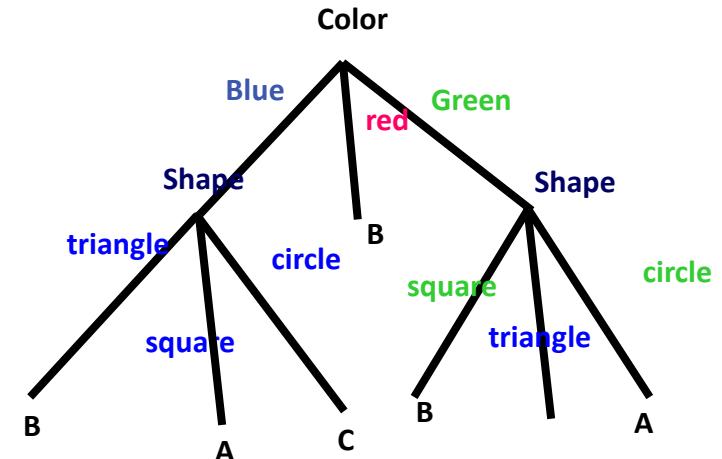
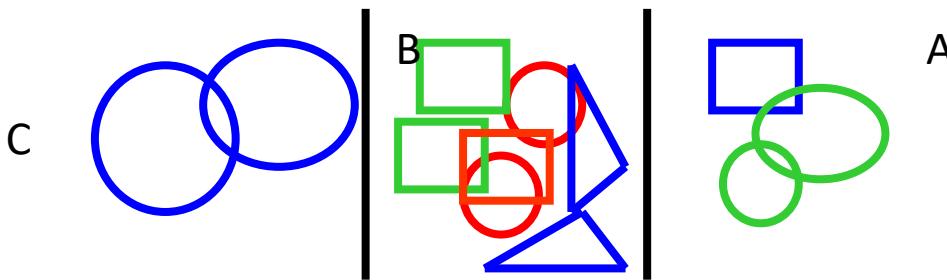


(Color=blue AND Shape=triangle \Rightarrow Label=B) AND
(Color=blue AND Shape=square \Rightarrow Label=A) AND
(Color=blue AND Shape=circle \Rightarrow Label=C) AND....

Learning a decision tree

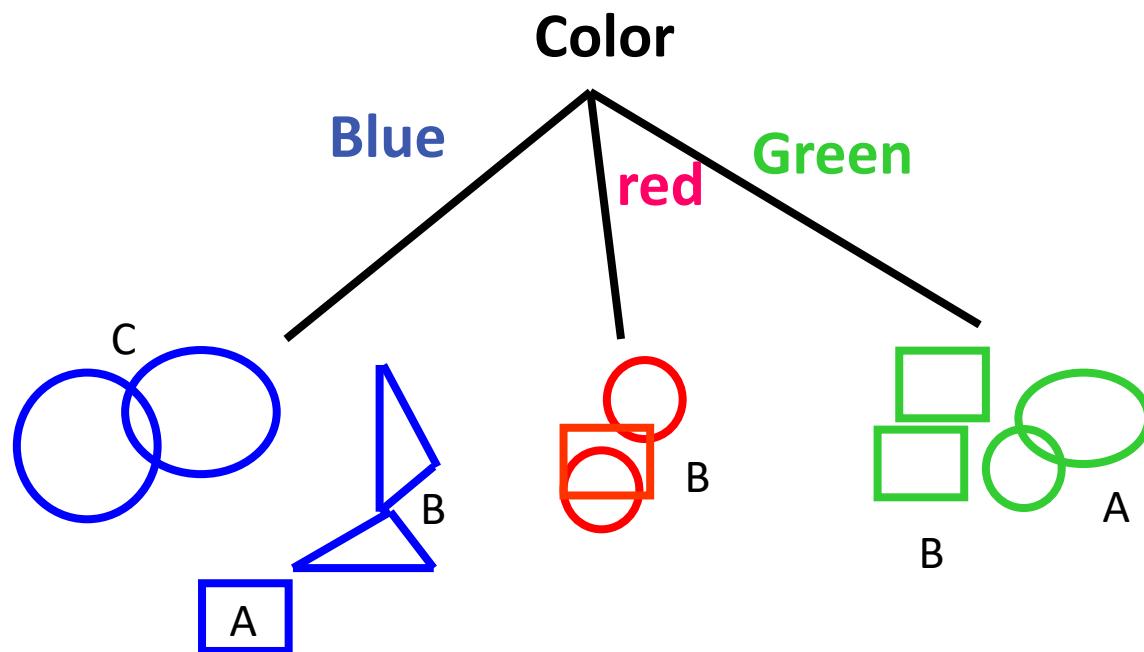
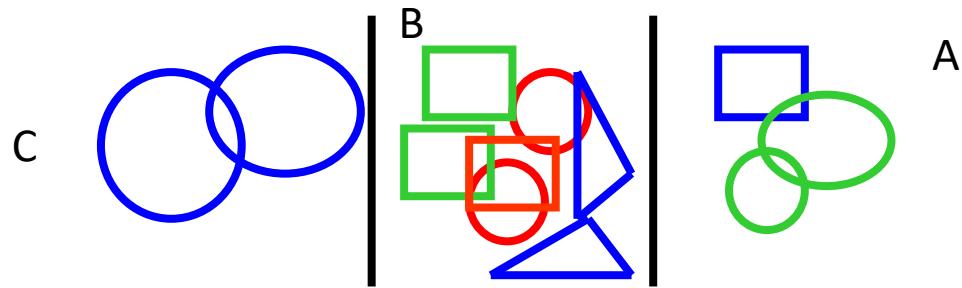
Basic Decision Trees Learning Algorithm

- ❖ Data is processed in Batch
(i.e. all the data available)
- ❖ Recursively build a decision tree top down.



DT algorithm: ID3(S , Attributes, Label)

- ❖ A recursive algorithm
- ❖ Recursively build a decision tree top down.
- ❖ Base case:
 - If all examples are labeled the same
 - Return a single node with the label
- Otherwise
 - Pick an attribute and create branches
 - Split the tree
 - (see next slide for details)



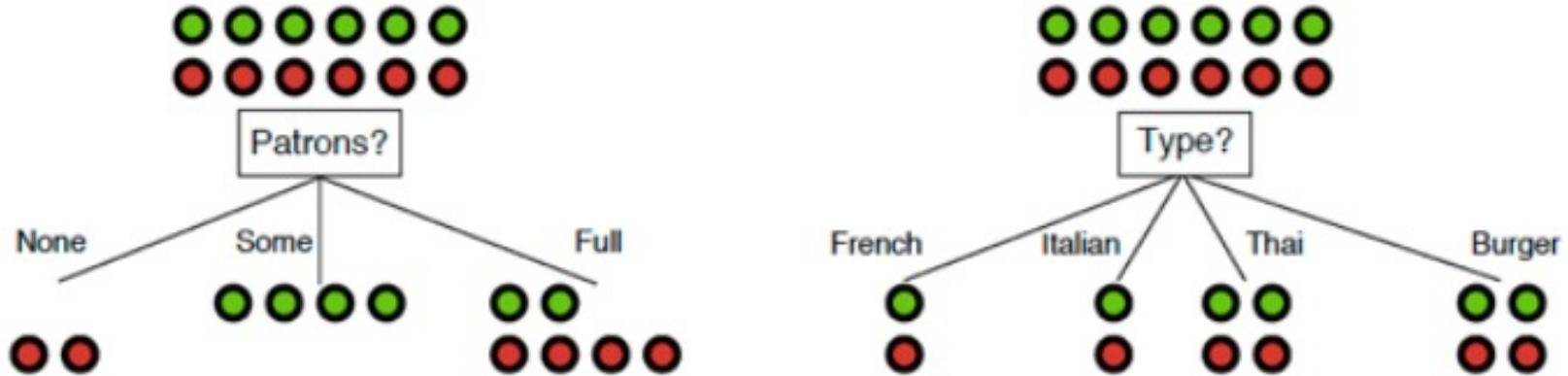
DT algorithm: ID3(S , Attributes, Label)

1. If all examples have a same label
return a single node tree with LabelBase case
2. $A = \text{attribute in Attributes that } \underline{\text{best}} \text{ classifies } S$
3. For each possible value v of A
 1. Add a new tree branch corresponding to $A=v$
 2. Let S_v be the subset of examples in S with $A=v$
 3. if S_v is empty:
add leaf node with the common value of Label in S
else: below this branch add the subtree
 $\text{ID3}(S_v, \text{Attributes} - \{A\}, \text{Label})$

Which attribute to split?

- ❖ The goal is to have the resulting decision tree as small as possible
- ❖ Finding the minimal decision tree consistent with the data is NP-hard
- ❖ A greedy heuristic search for a simple tree (cannot guarantee optimality)

Which attribute to split?



Patrons? is a better choice—gives **information** about the classification

How to quantify it?

The most popular heuristics is based on **information gain**

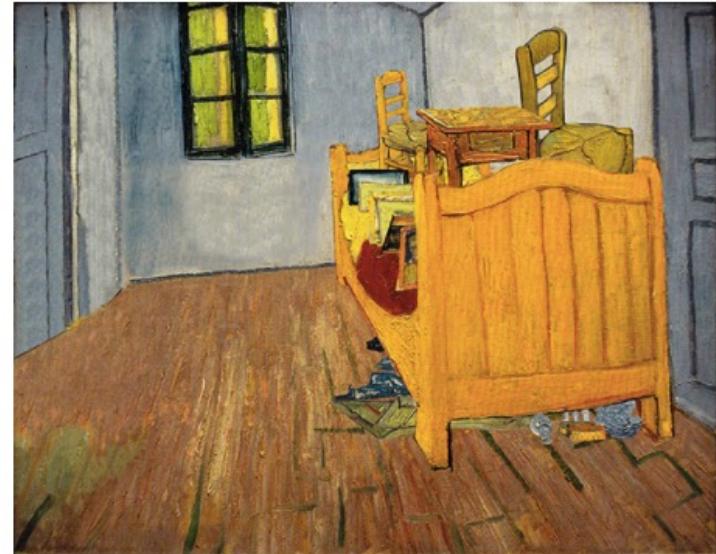
How to measure information gain?

- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by entropy



Vincent Van Gogh: Bedroom in Arles

High entropy



By Ursus Wehrli

Low entropy

How to measure information gain?

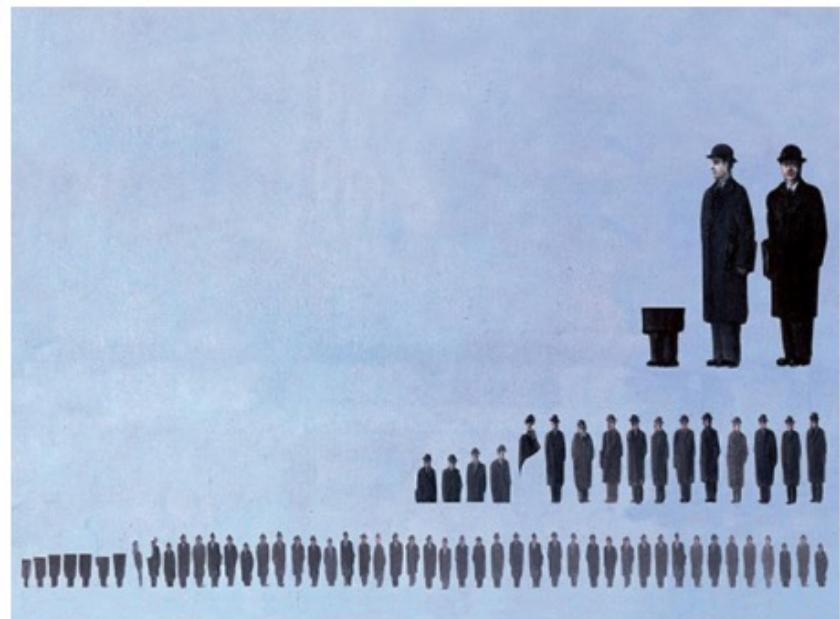
- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by Entropy

René Magritte "Golconda"



High entropy

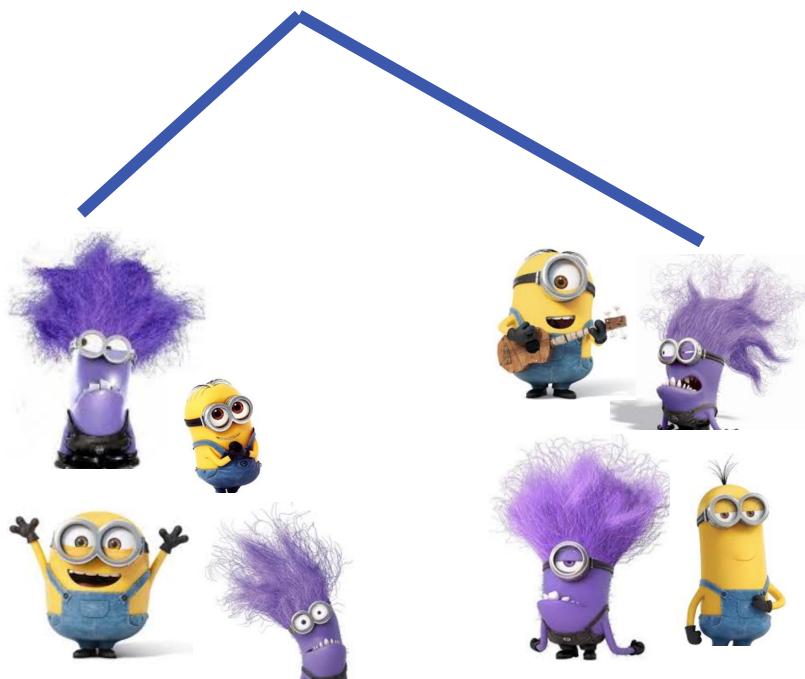
Bv Ursus Wehrli



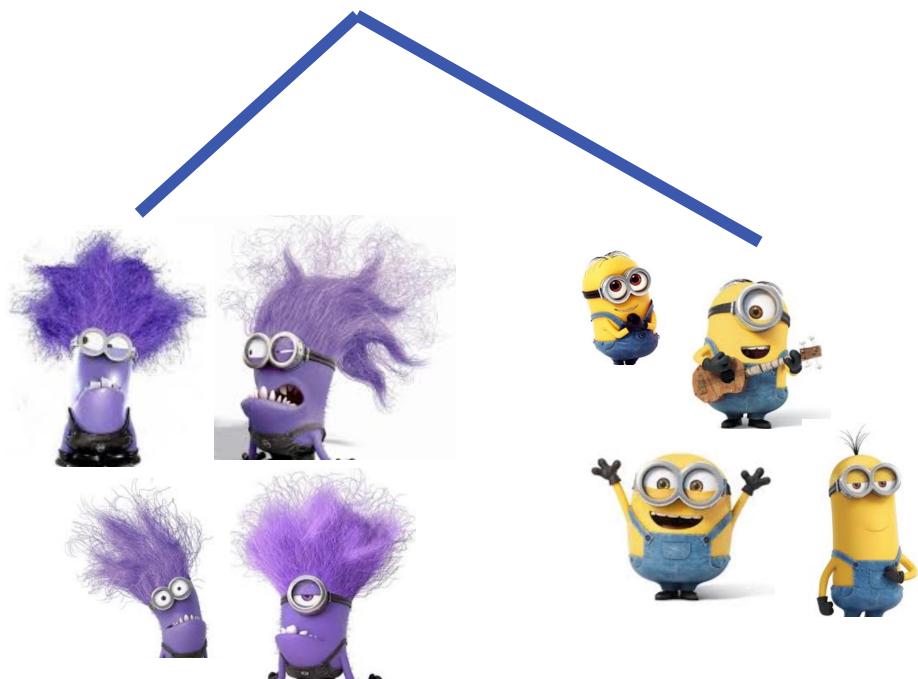
Low entropy

How to measure information gain?

- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by Entropy



High entropy



Low entropy

Entropy

- ❖ Entropy (impurity, disorder) of a set of examples, S , relative to a binary classification is:

$$H[S] = -P_+ \log_2(P_+) - P_- \log_2(P_-)$$

- ❖ where P_+ is the proportion of positive examples in S and P_- is the proportion of negatives.

Here we define $0 \log 0 = 0$

Entropy (formal definition)

- ❖ If a random variable S has K different values, $a_1, a_2, \dots a_K$, its entropy is given by

$$H[S] = - \sum_{v=1}^K P(S = a_v) \log_2 P(S = a_v)$$



$$H[S] = -\frac{1}{2} \log_2 \left(\frac{1}{2}\right) - \frac{1}{2} \log_2 \left(\frac{1}{2}\right) = 1$$

$$H[S] = -\frac{1}{1} \log_2 (1) = 0$$

Entropy (intuition)

- ❖ In average, how many bits do we need to send the message (#bits/#length of message)



Entropy (intuition)

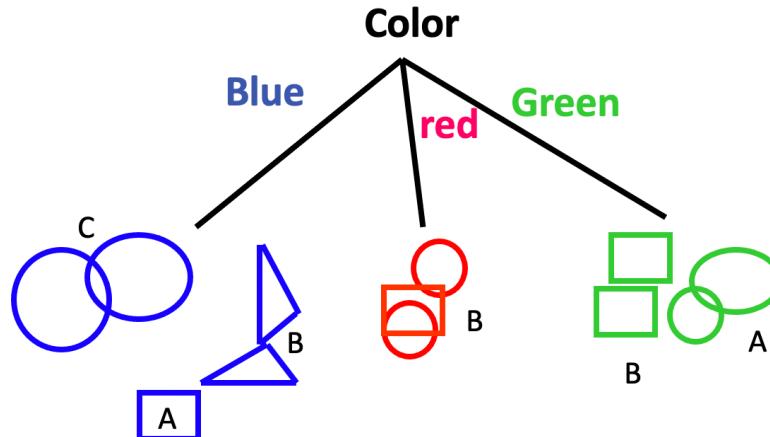
- ❖ In average, how many bits do we need to send the message (#bits/#length of message)
- ❖ Consider you have four possible tokens (a,b,c,d). What is the best way to encode them?
- ❖ All examples belong to the same category
 - e.g., aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 - no need to communicate (or just 1 bit)
- ❖ If all the examples are equally mixed (0.25, 0.25, 0.25, 0.25):
 - e.g., abbacaccddd.....
 - two bits for each token: (a:00, b:01, c:10, d:11)
- ❖ If $\frac{1}{4}$ of message is a, and $\frac{1}{2}$ is b and $\frac{1}{4}$ is c in average:
 - e.g., abbbbacc.....
 - (a:00, b:1, c:01, d:--)

Information Gain

- The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- S_v is the subset of S for which attribute a has value v .
- The entropy of partitioning the data is calculated by weighing the entropy of each partition by its size



Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(strong),
W(eak)

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Current entropy:

$$p = 9/14$$

$$n = 5/14$$

$$H(\text{Play?}) = -(9/14) \log_2(9/14) \\ -(5/14) \log_2(5/14)$$

$$\approx 0.94$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Outlook = overcast: 4 of 14 examples

$$p = 4/4 \quad n = 0 \quad H_o = 0$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Outlook = overcast: 4 of 14 examples

$$p = 4/4 \quad n = 0 \quad H_o = 0$$

Outlook = rainy: 5 of 14 examples

$$p = 3/5 \quad n = 2/5 \quad H_R = 0.971$$

Expected entropy:

$$(5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 \\ = 0.694$$

Information gain:

$$0.940 - 0.694 = 0.246$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Information Gain: Humidity

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Humidity = High:

$$p = 3/7 \quad n = 4/7$$

$$H_h = 0.985$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	H	W
2	S	H	H	S
3	O	H	H	W
4	R	M	H	W
5	R	C	N	W
6	R	C	N	S
7	O	C	N	S
8	S	M	H	W
9	S	C	N	W
10	R	M	N	W
11	S	M	N	S
12	O	M	H	S
13	O	H	N	W
14	R	M	H	S

Humidity = High:

$$p = 3/7 \quad n = 4/7 \quad H_h = 0.985$$

Humidity = Normal:

$$p = 6/7 \quad n = 1/7 \quad H_o = 0.592$$

Expected entropy:

$$(7/14) \times 0.985 + (7/14) \times 0.592 = 0.7885$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Humidity = High:

$$p = 3/7 \quad n = 4/7 \quad H_h = 0.985$$

Humidity = Normal:

$$p = 6/7 \quad n = 1/7 \quad H_o = 0.592$$

Expected entropy:

$$(7/14) \times 0.985 + (7/14) \times 0.592 = 0.7885$$

Information gain:

$$0.940 - 0.7885 = 0.1515$$

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246

Humidity: 0.151

Wind: 0.048

Temperature: 0.029

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246

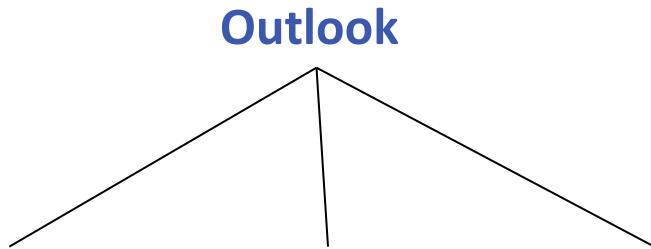
Humidity: 0.151

Wind: 0.048

Temperature: 0.029

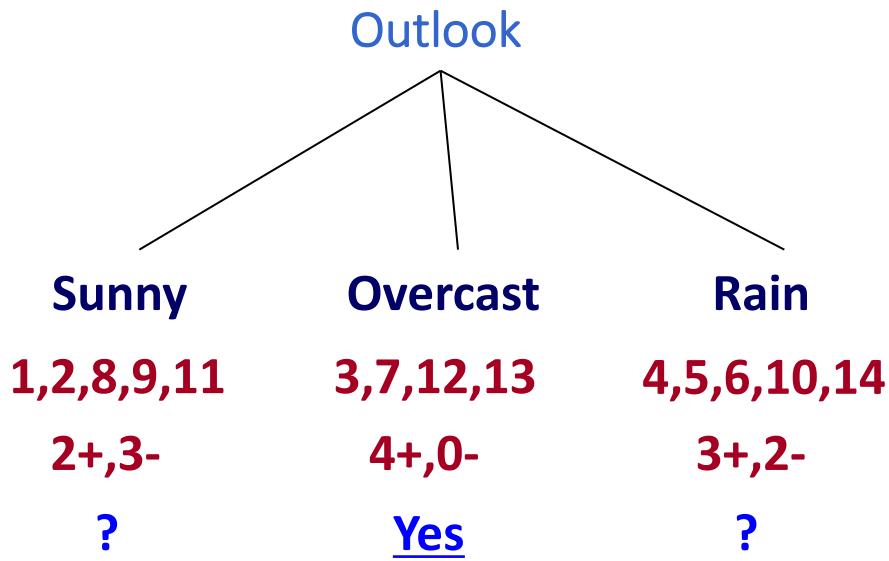
→ Split on Outlook

An Illustrative Example



Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example

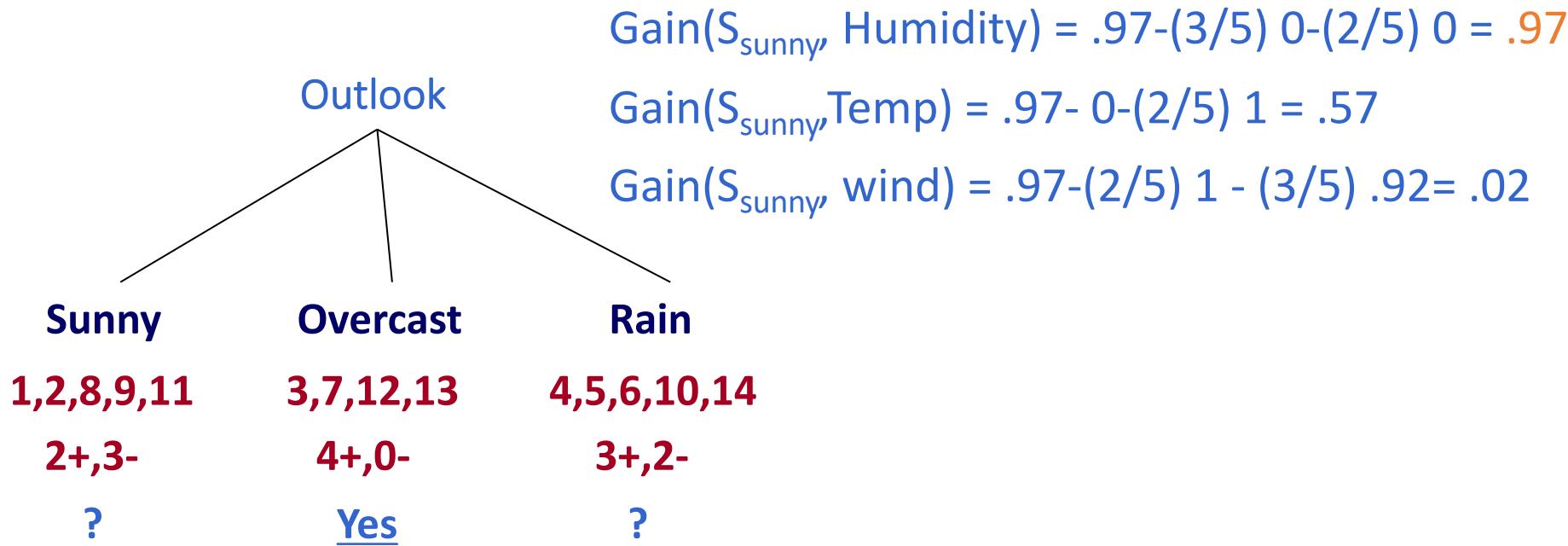


Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

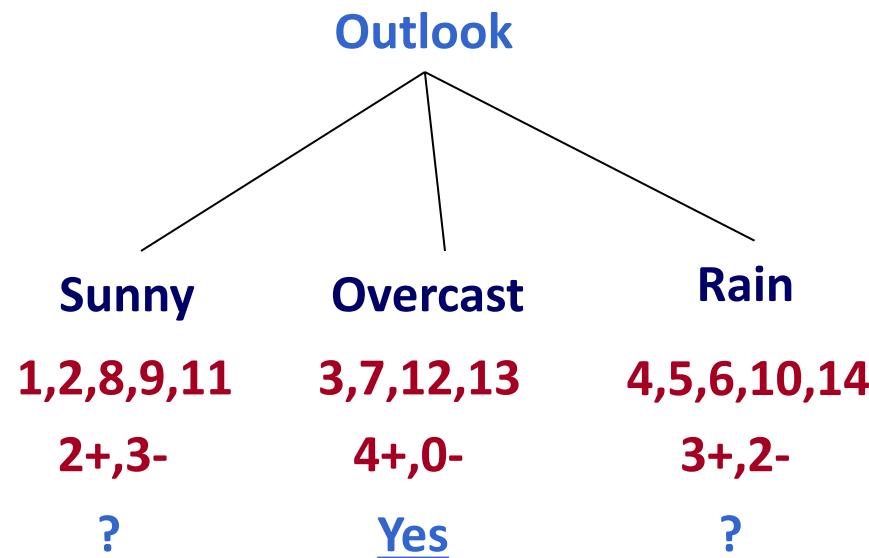
	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example

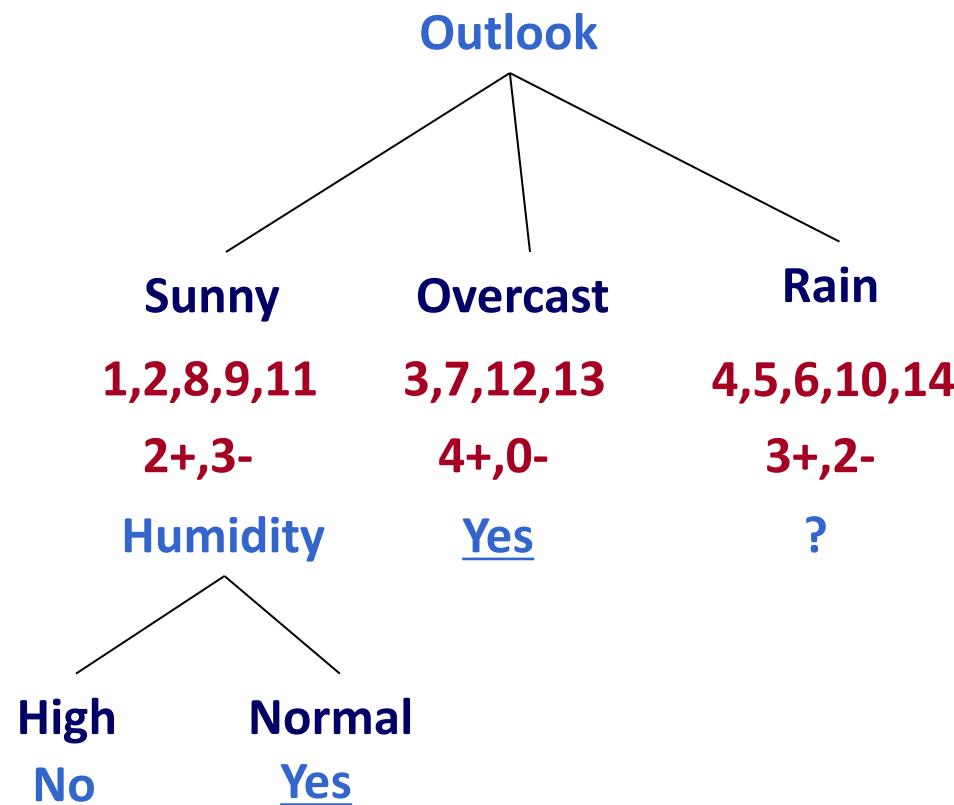


Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

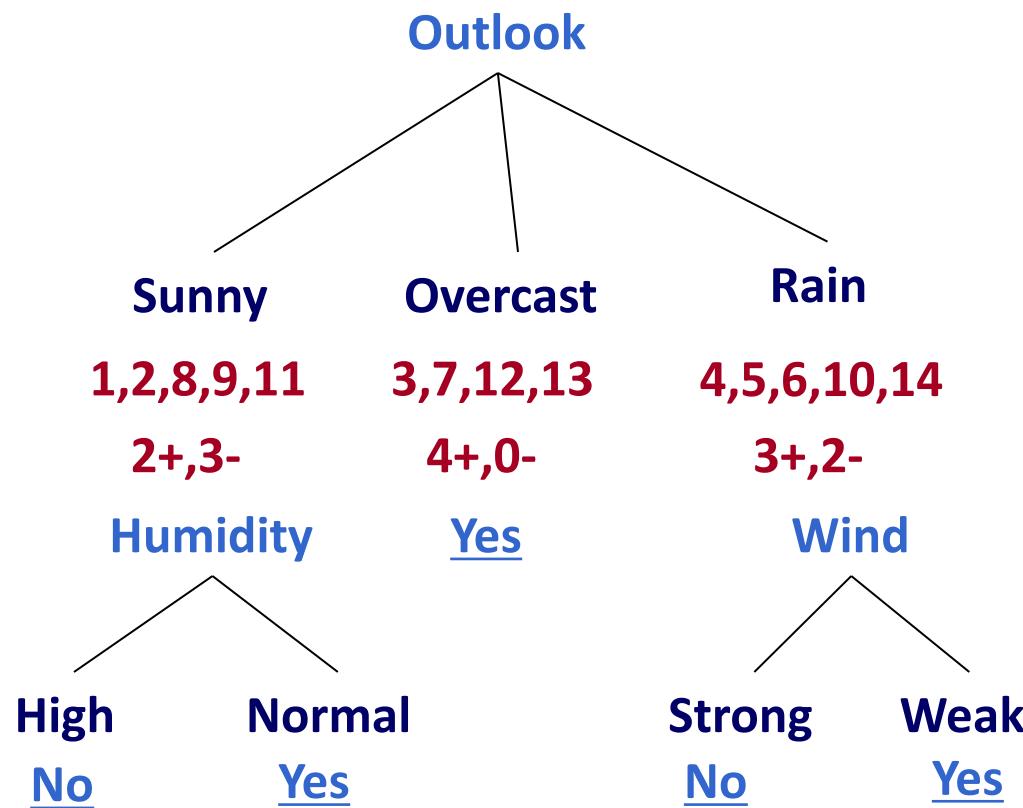
An Illustrative Example



An Illustrative Example



An Illustrative Example



Summary: Learning Decision Trees

1. **Representation**: What are decision trees?

- ❖ A hierarchical data structure that represents data

2. **Algorithm**: Learning decision trees

The ID3 algorithm: A greedy heuristic

- ❖ If all the examples have the same label, create a leaf with that label
- ❖ Otherwise, find the “most informative” attribute and split the data for different values of that attribute
- ❖ Recurse on the splits

Linear Models

Fall 2022

Kai-Wei Chang

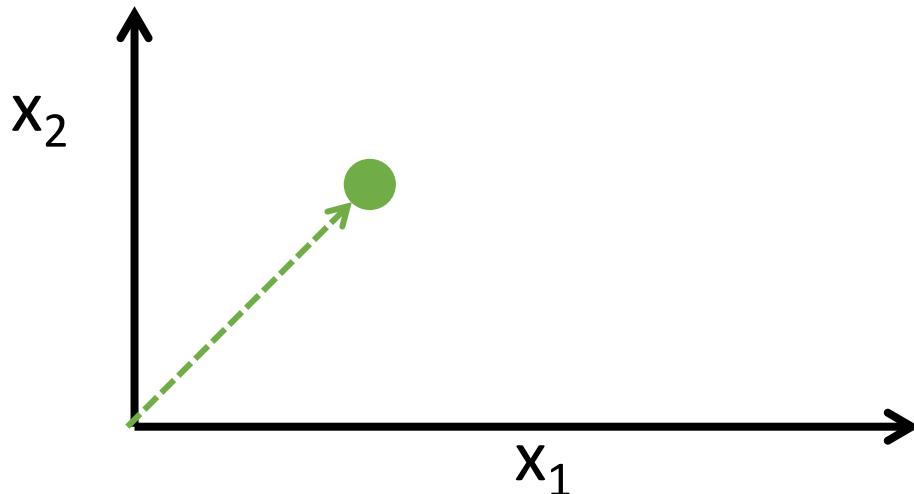
CS @ UCLA

kw+cm146@kwchang.net

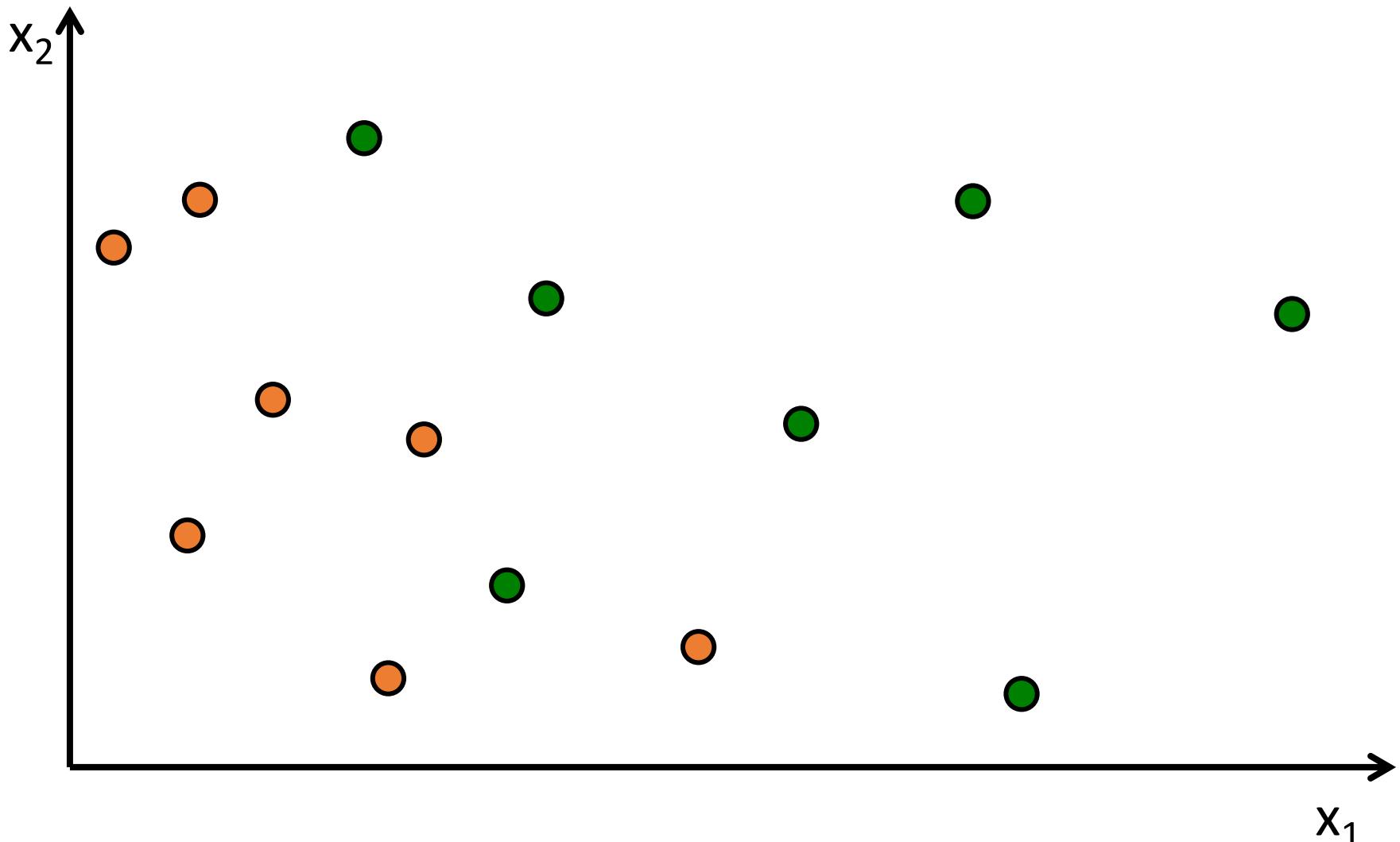
The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Recap: \mathcal{X} as a vector space

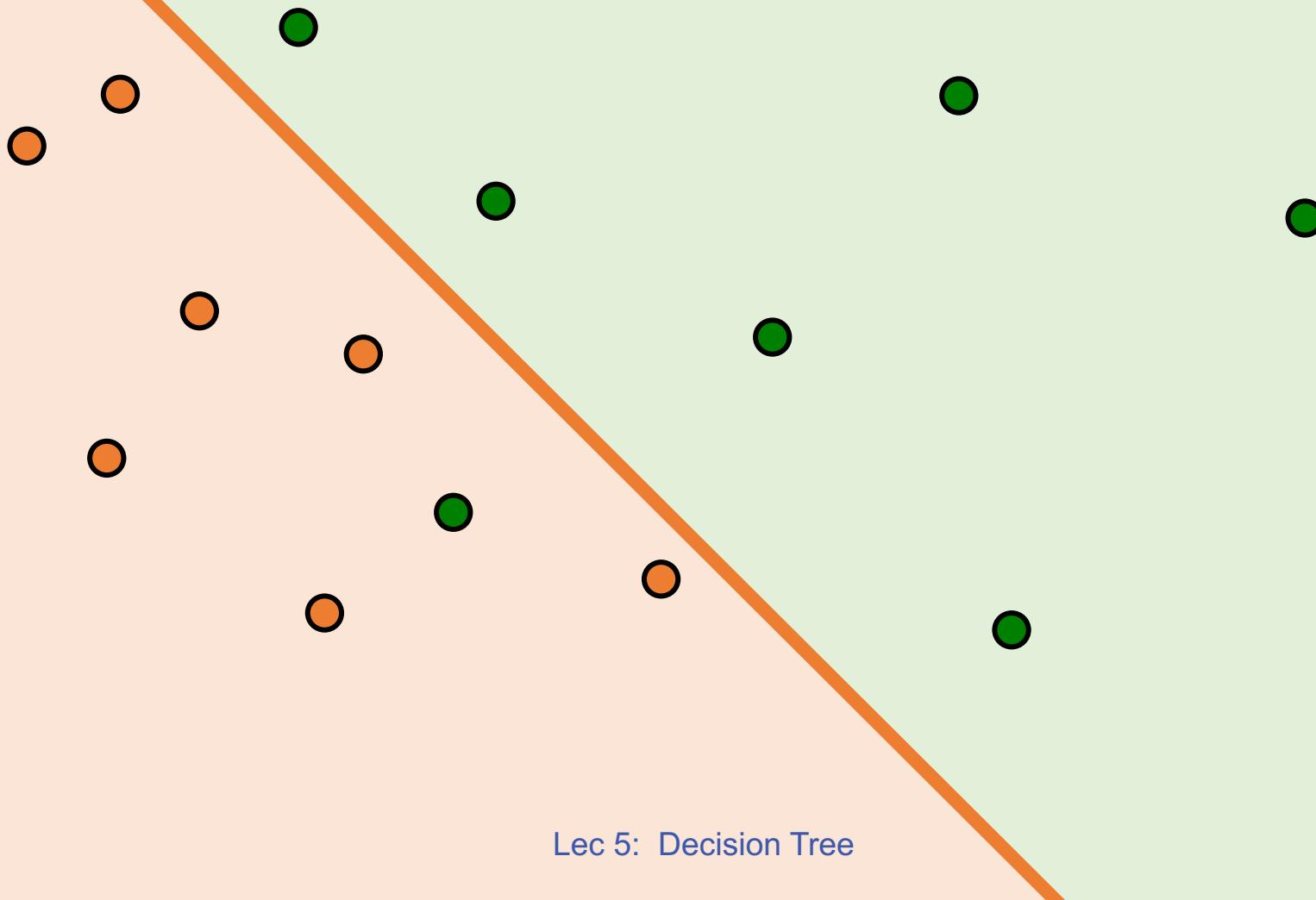
- ❖ \mathcal{X} is an N-dimensional vector space (e.g. \mathbb{R}^N)
 - ❖ Each dimension = one feature.
- ❖ Each \mathbf{x} is a **feature vector** (hence the boldface \mathbf{x}).
- ❖ Think of $\mathbf{x} = [x_1 \dots x_N]$ as a point in \mathcal{X} :



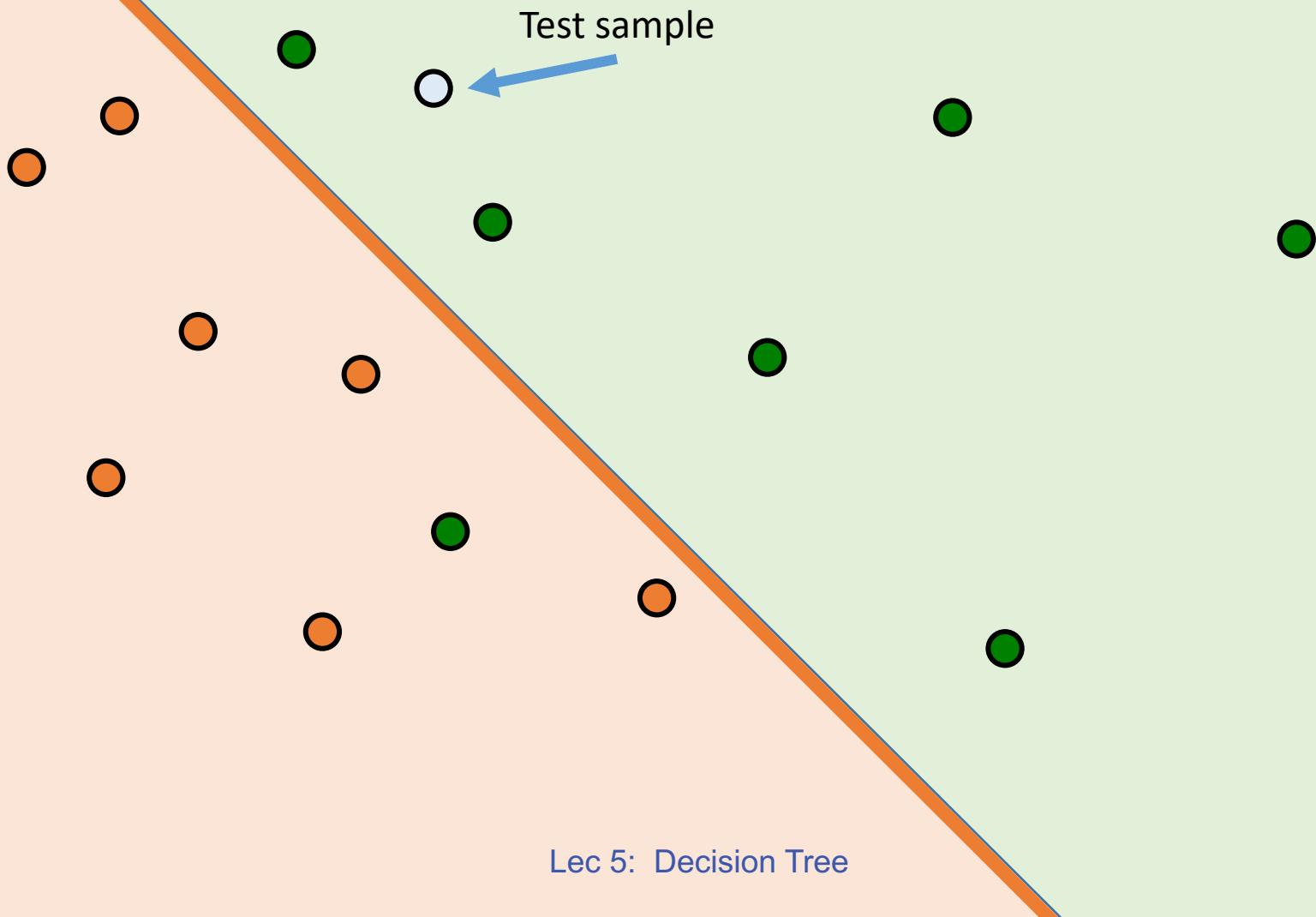
Training data



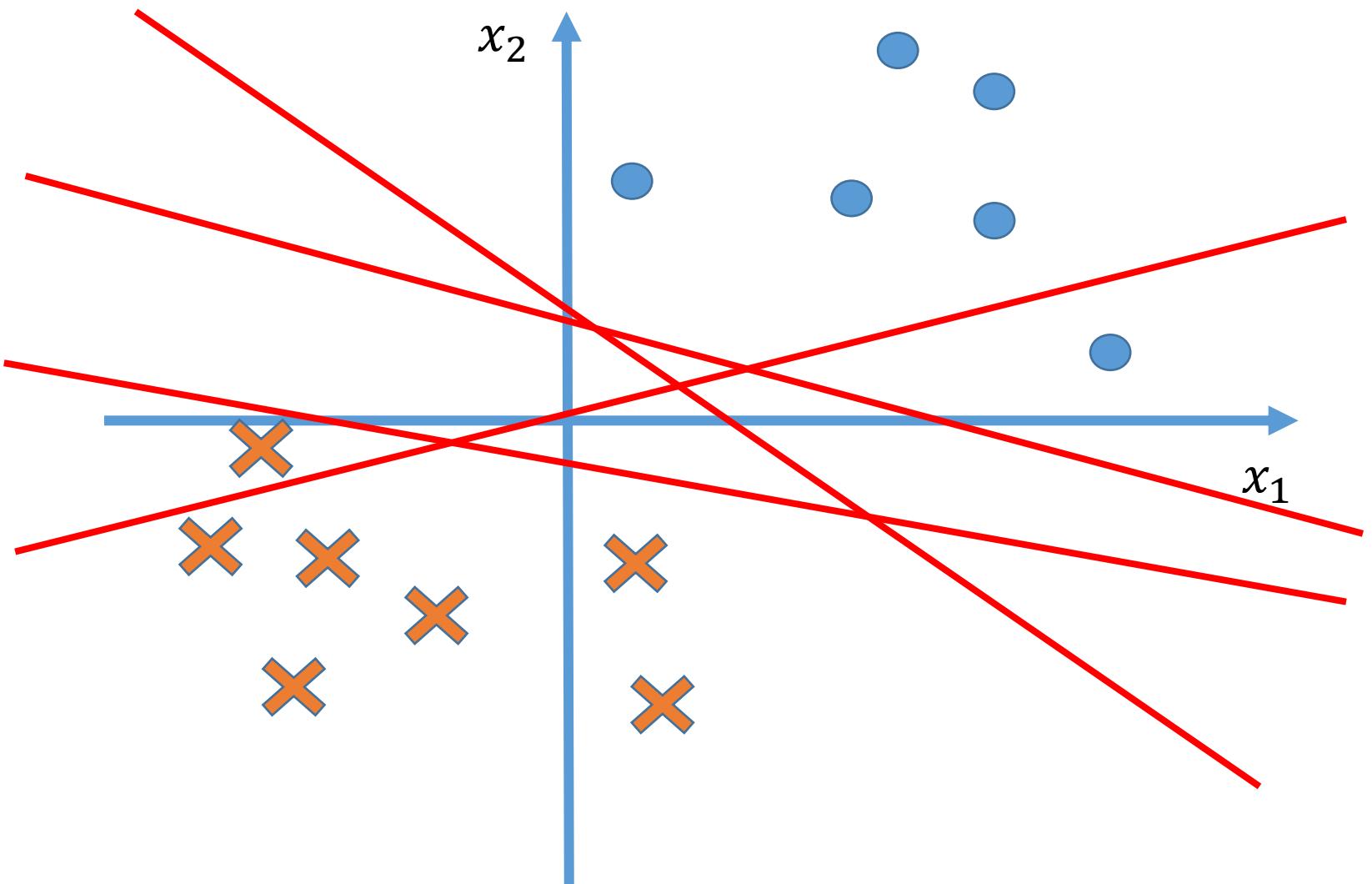
Hyperplane Separates the Space



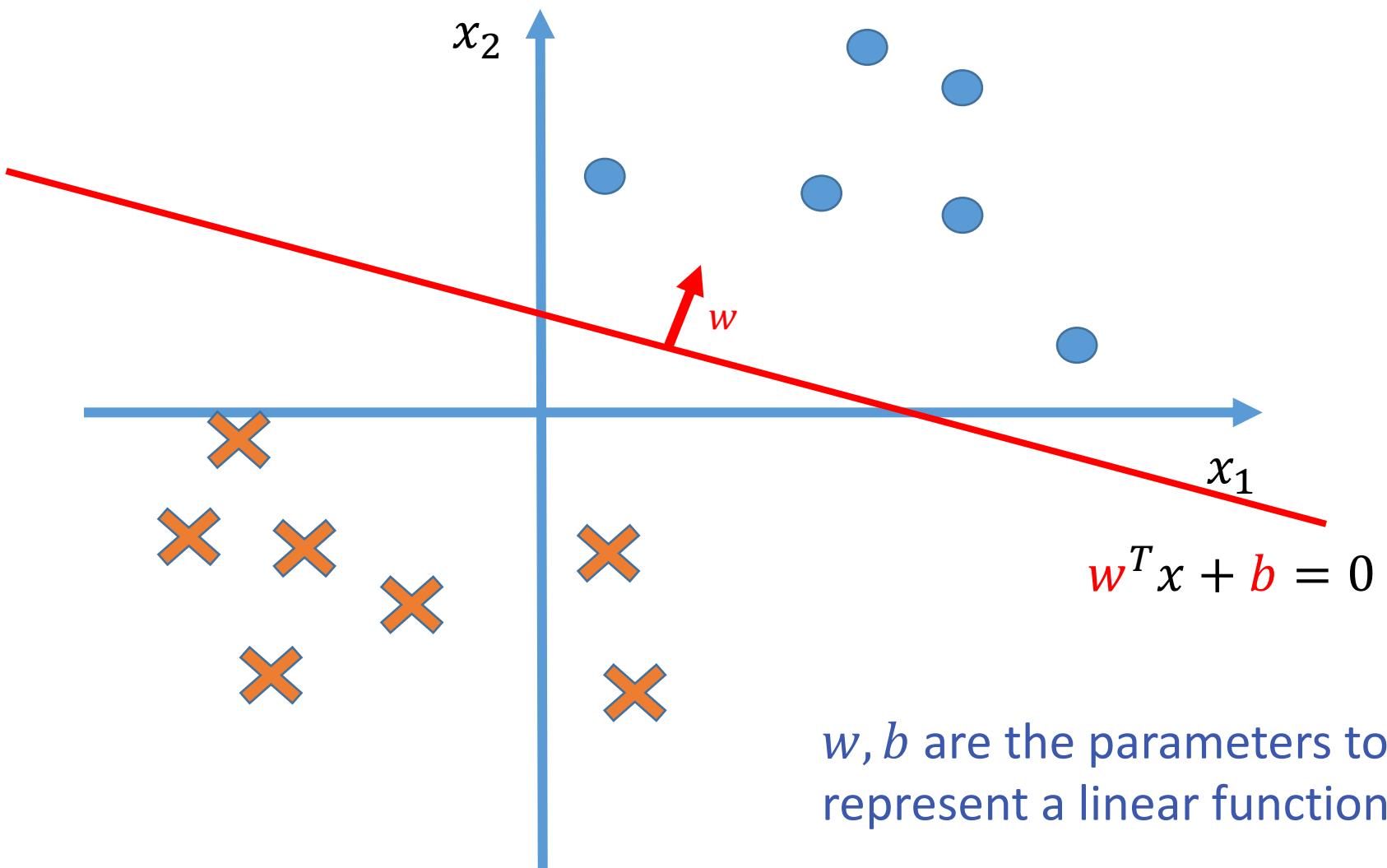
Test Phase



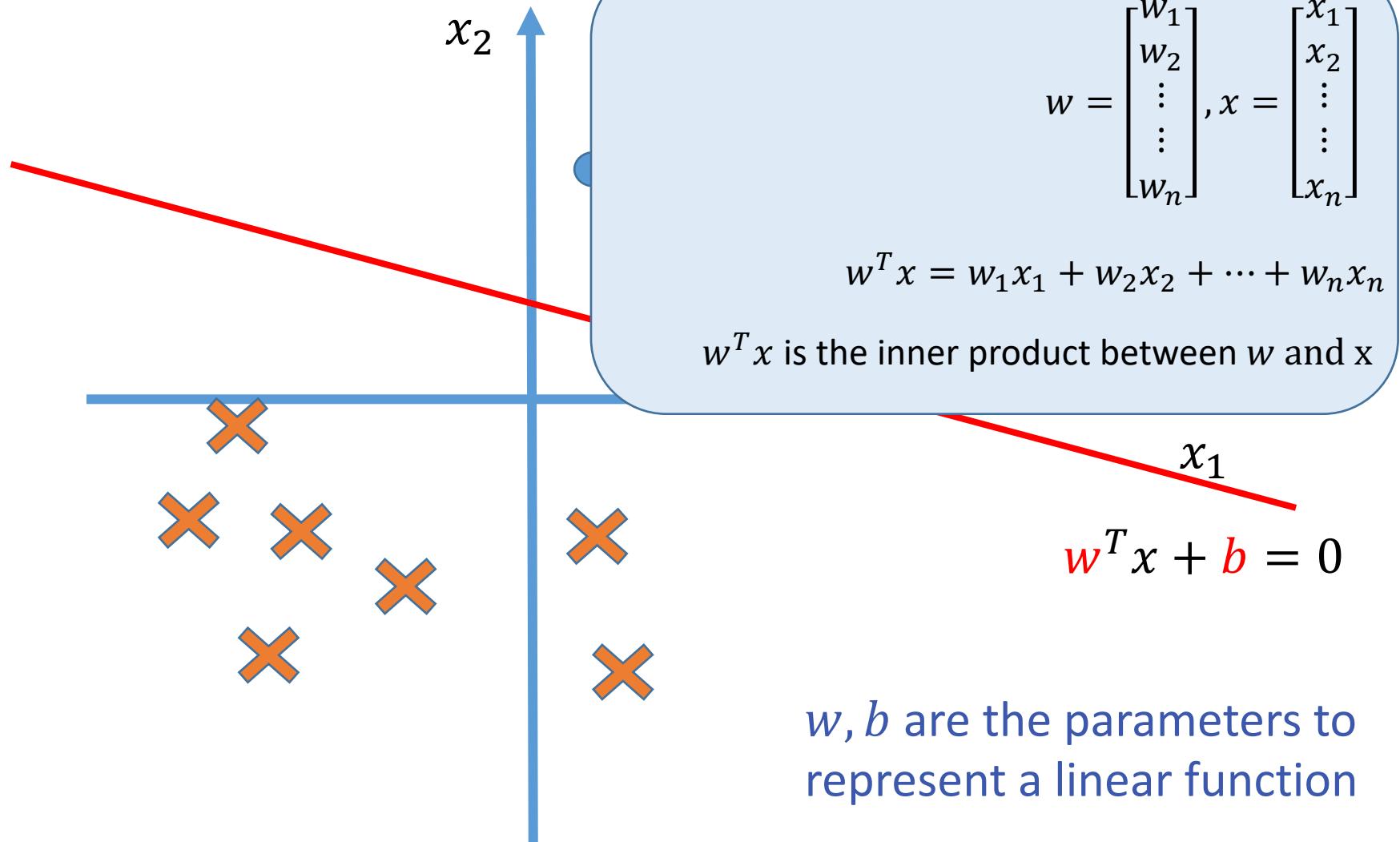
Hypothesis space: linear model



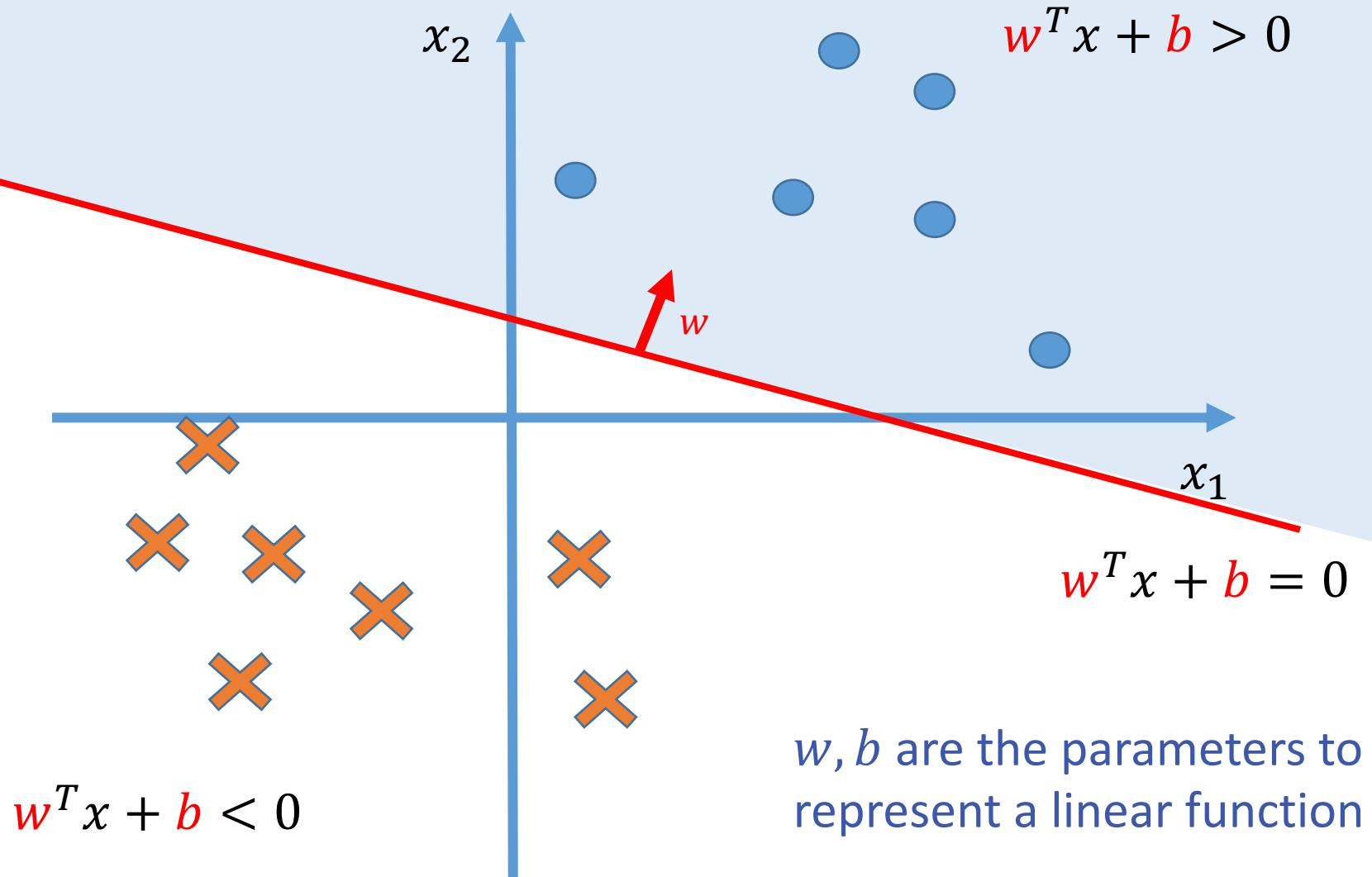
Hypothesis space: linear model



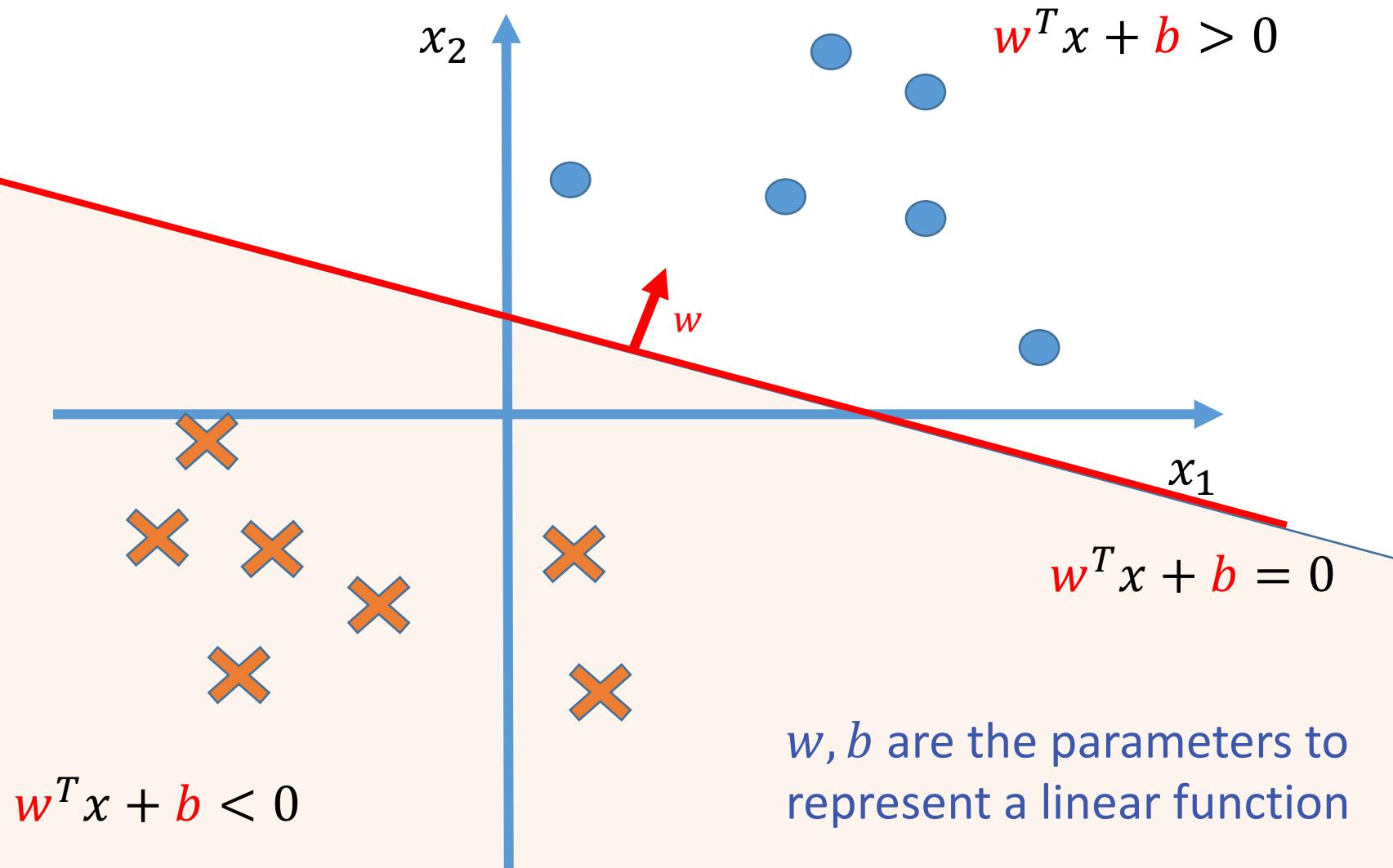
Hypothesis space: linear model



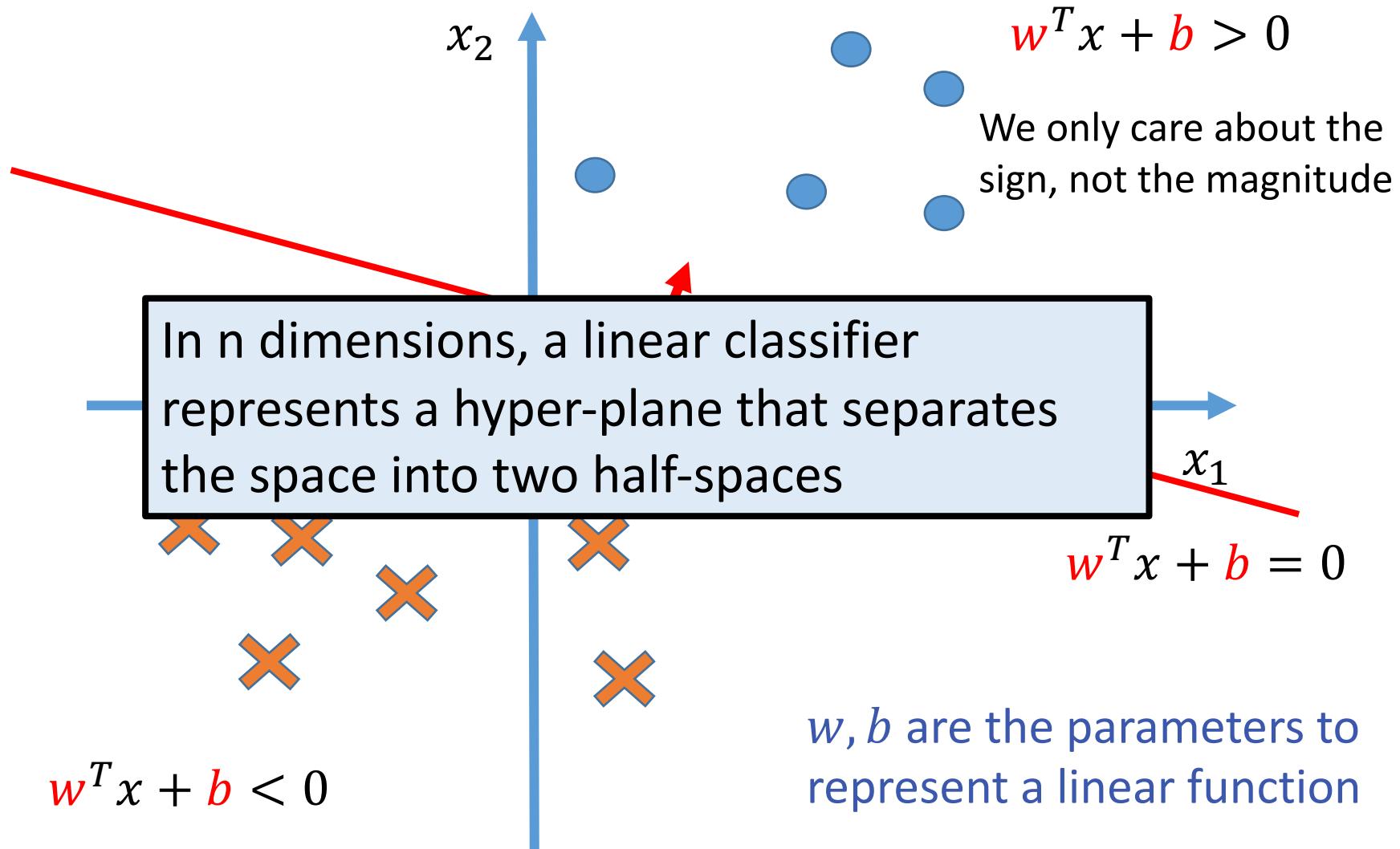
Hypothesis space: linear model



Hypothesis space: linear model

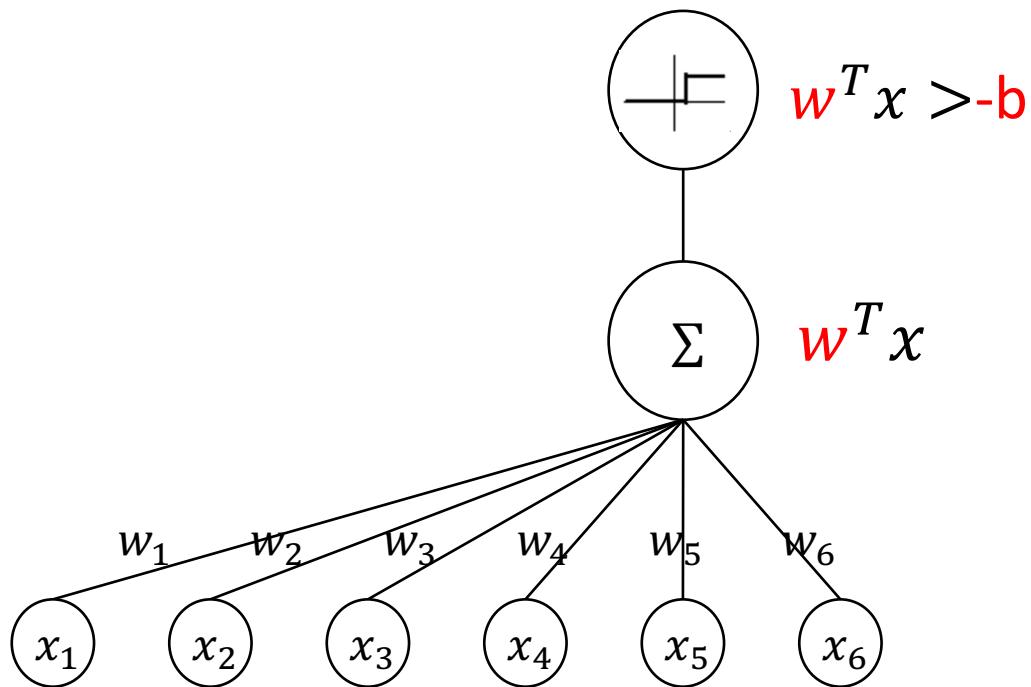


Hypothesis space: linear model



Recall: Linear Classifiers

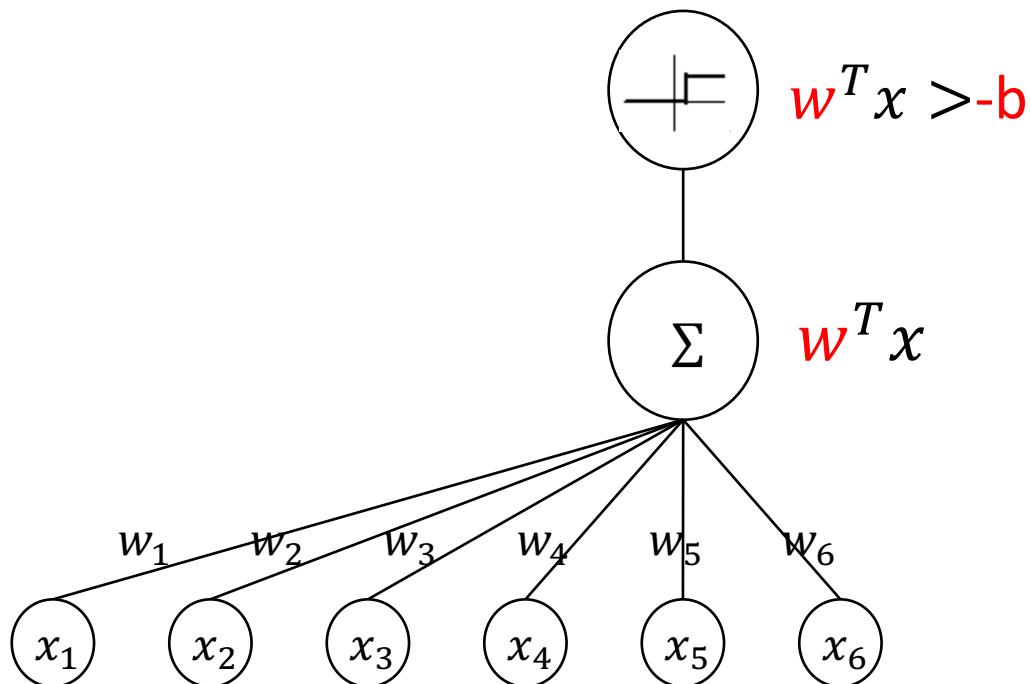
- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

Recall: Linear Classifiers

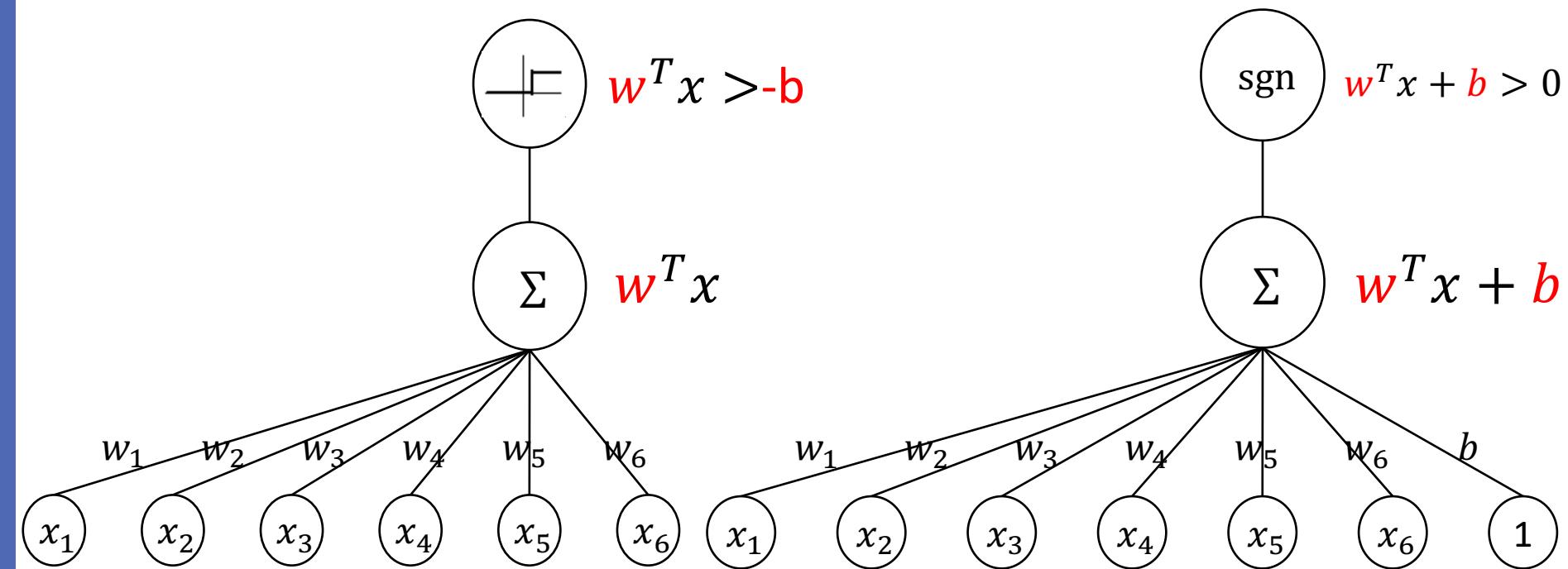
- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

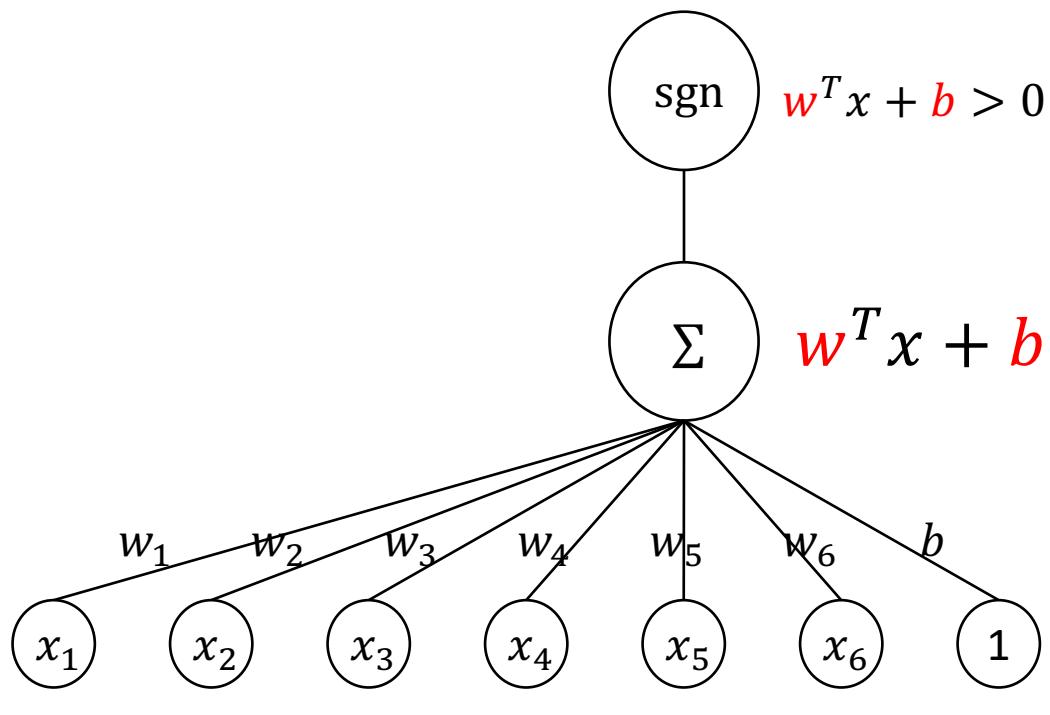
Recall: Linear Classifiers

- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

A simple trick to remove the bias term b

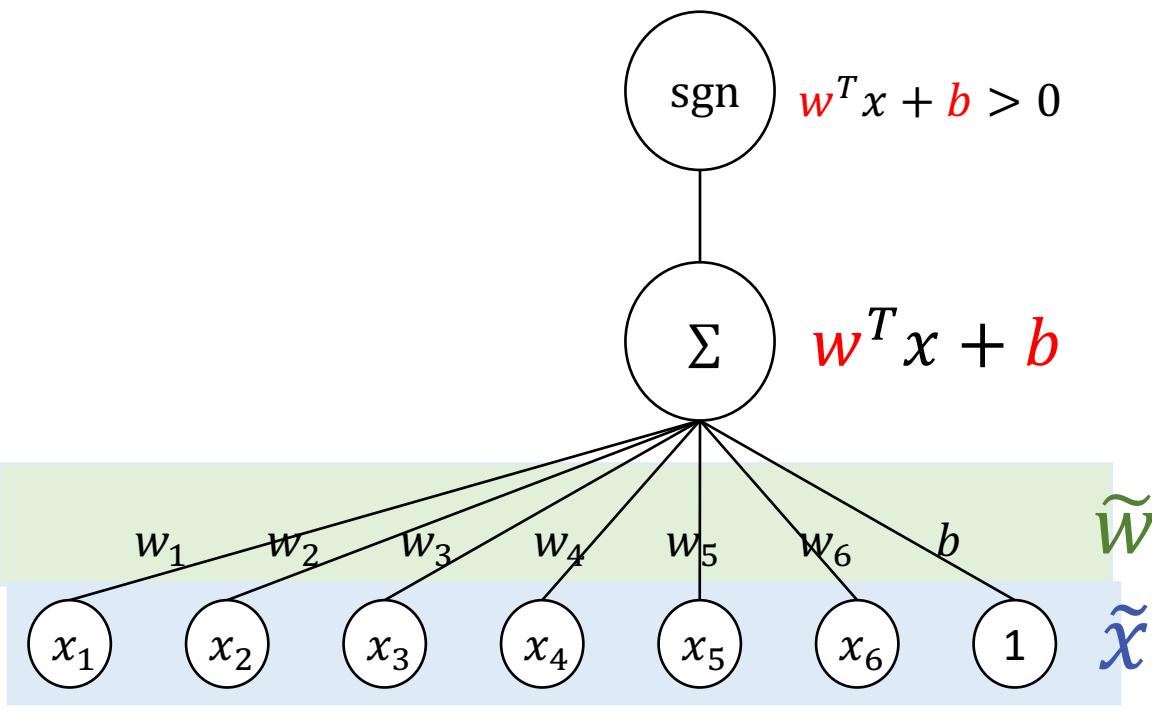


$$\begin{aligned}w^T x + b \\= [w^T \ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\= \tilde{w} \cdot \tilde{x}\end{aligned}$$

$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

A simple trick to remove the bias term b

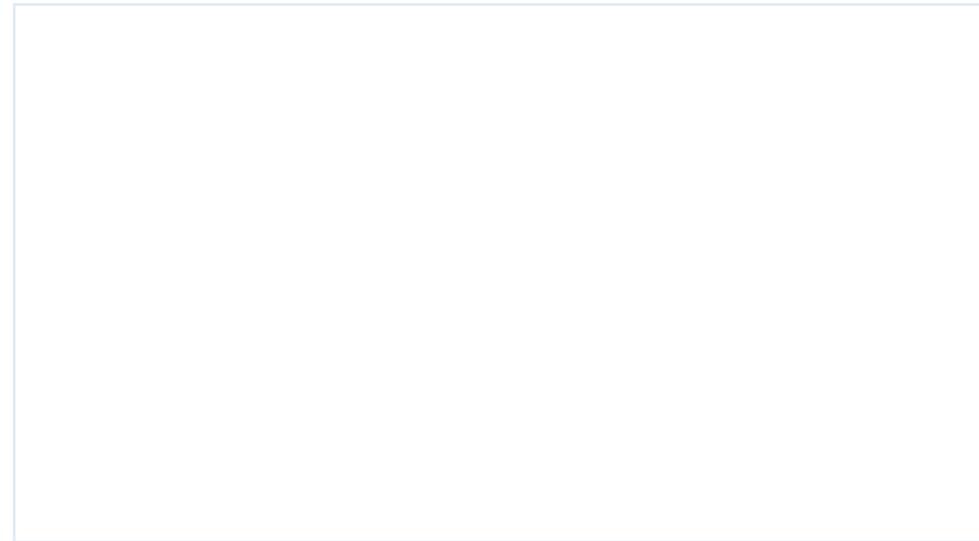
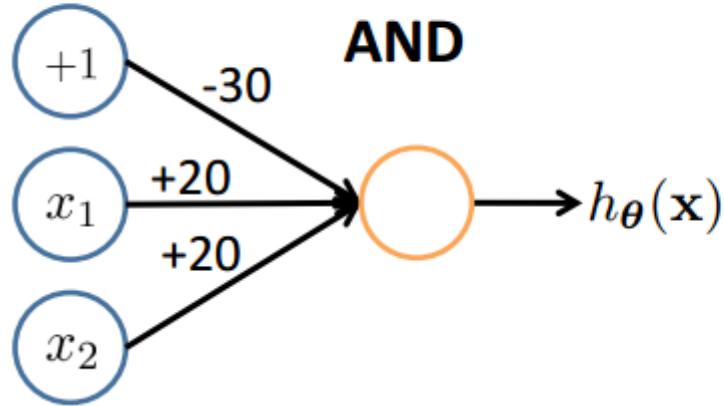


$$\begin{aligned}w^T x + b \\= [w^T \ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\= \tilde{w} \cdot \tilde{x}\end{aligned}$$

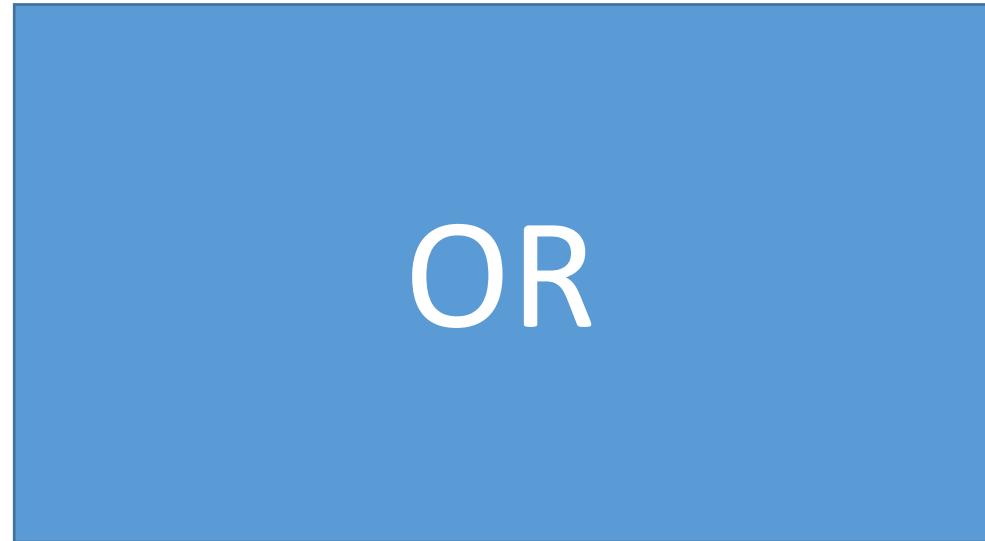
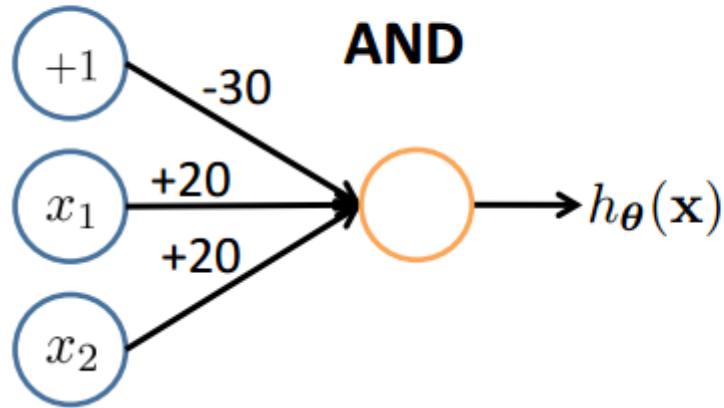
$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

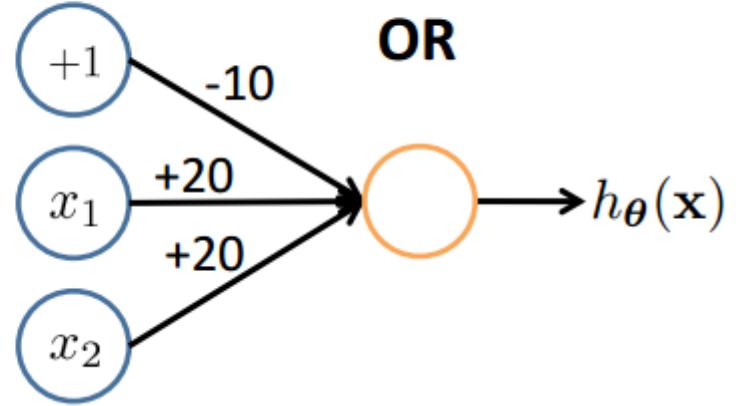
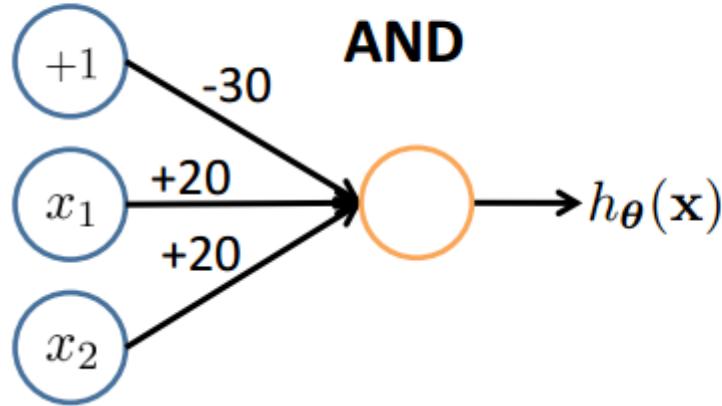
Representing Boolean Functions



Representing Boolean Functions



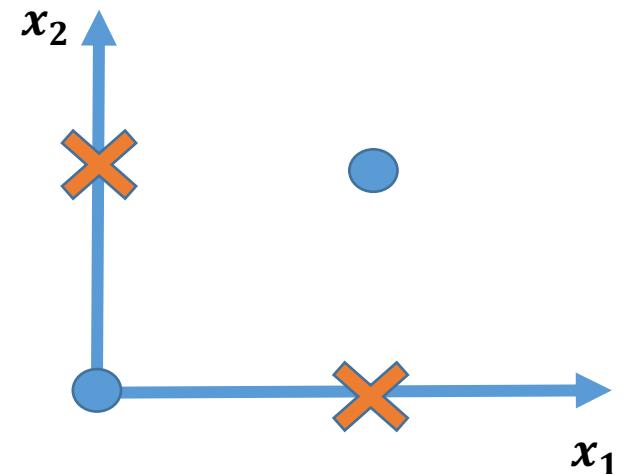
Representing Boolean Functions



Limitation

- ❖ Can linear model represent XNOR ?

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1

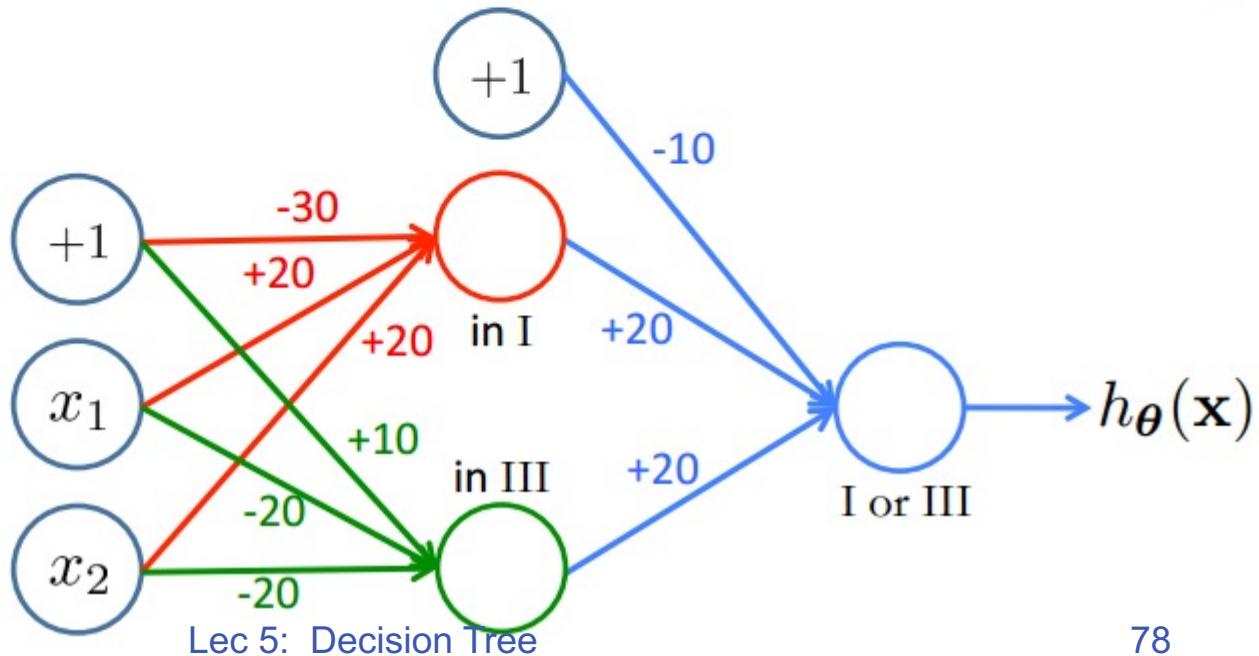
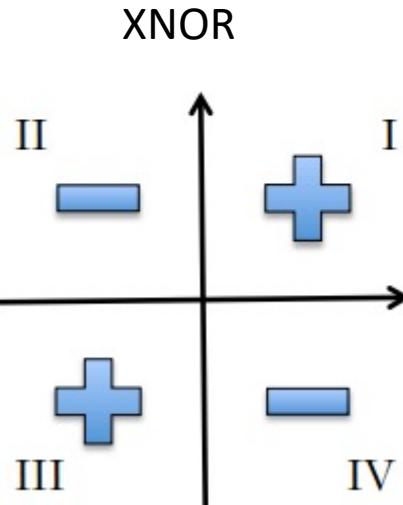
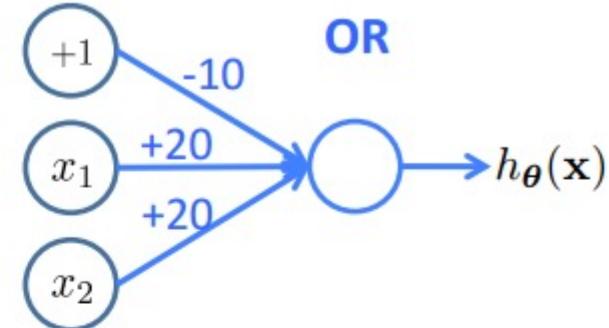
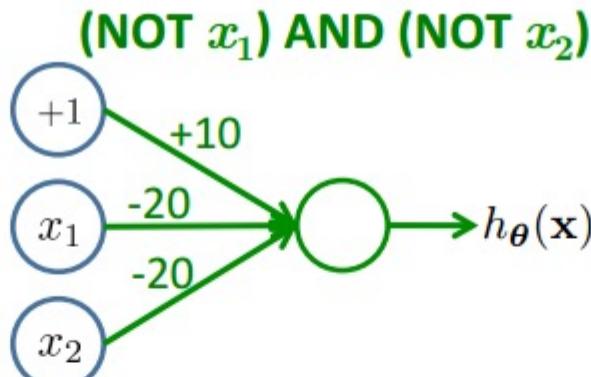
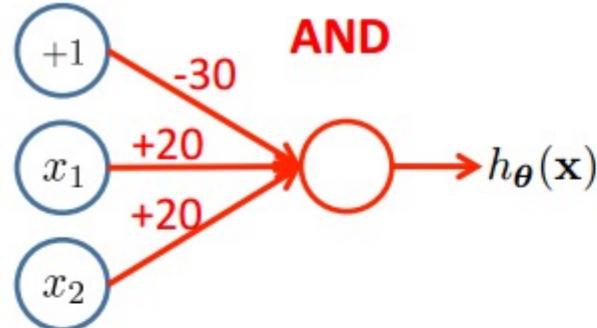


Assume the separating hyper plane is $w_1x_1 + w_2x_2 + b = 0$

From the four points we have:

$$\begin{aligned} w_1 + b &< 0 \\ w_2 + b &< 0 \\ b \geq 0 \\ w_1 + w_2 + b &\geq 0 \end{aligned} \quad \left. \begin{array}{l} w_1 + b < 0 \\ w_2 + b < 0 \\ w_1 + w_2 + b \geq 0 \end{array} \right\} \quad \left. \begin{array}{l} w_1 + b < 0 \\ w_2 + b < 0 \end{array} \right\} \quad w_1 + w_2 + b < 0$$

Multi-layer Perceptron (NN)



Lec 5: Decision Tree

Learning a Linear Classifier

- ❖ There are several algorithms/models
 - ❖ Perceptron
 - ❖ Logistic Regression
 - ❖ (Linear) Support Vector Machines
 - ❖ ...
- ❖ Based on different assumptions, you get different linear models

Lecture 6: Linear Model & Perceptron Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

- ❖ Quiz 1 will be due Today!!
- ❖ Hw1 update (please see the pinned message at Piazza)

```
84
```

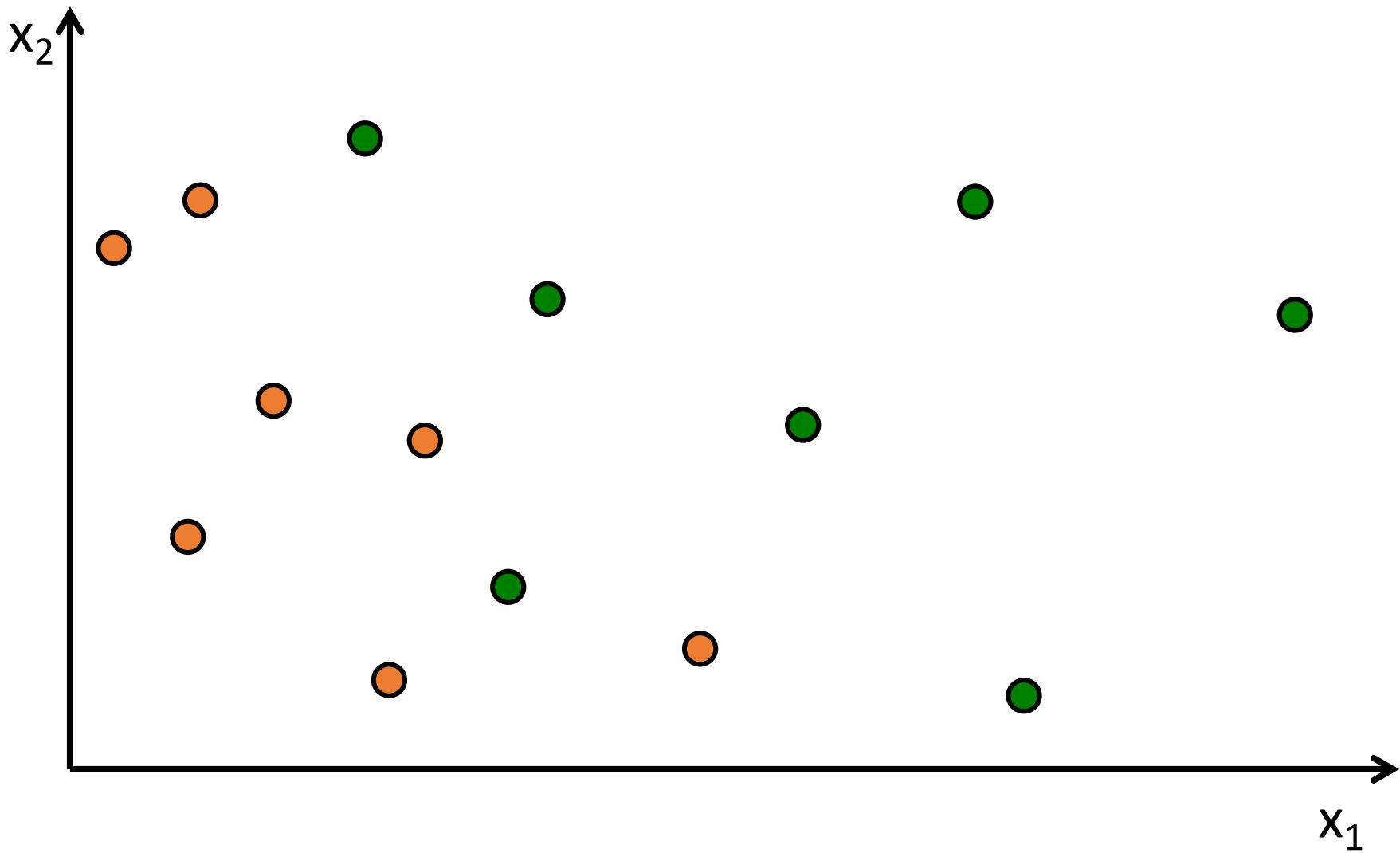
```
85 # Shuffle the data for cross-validation
86 import random
87 idx = list(range(n))
88 random.shuffle(idx)
89 X = np.take(X, idx, axis=0)
90 y = np.take(y, idx, axis=0)
```

anks, Wenda Fu, for identifying these issues.

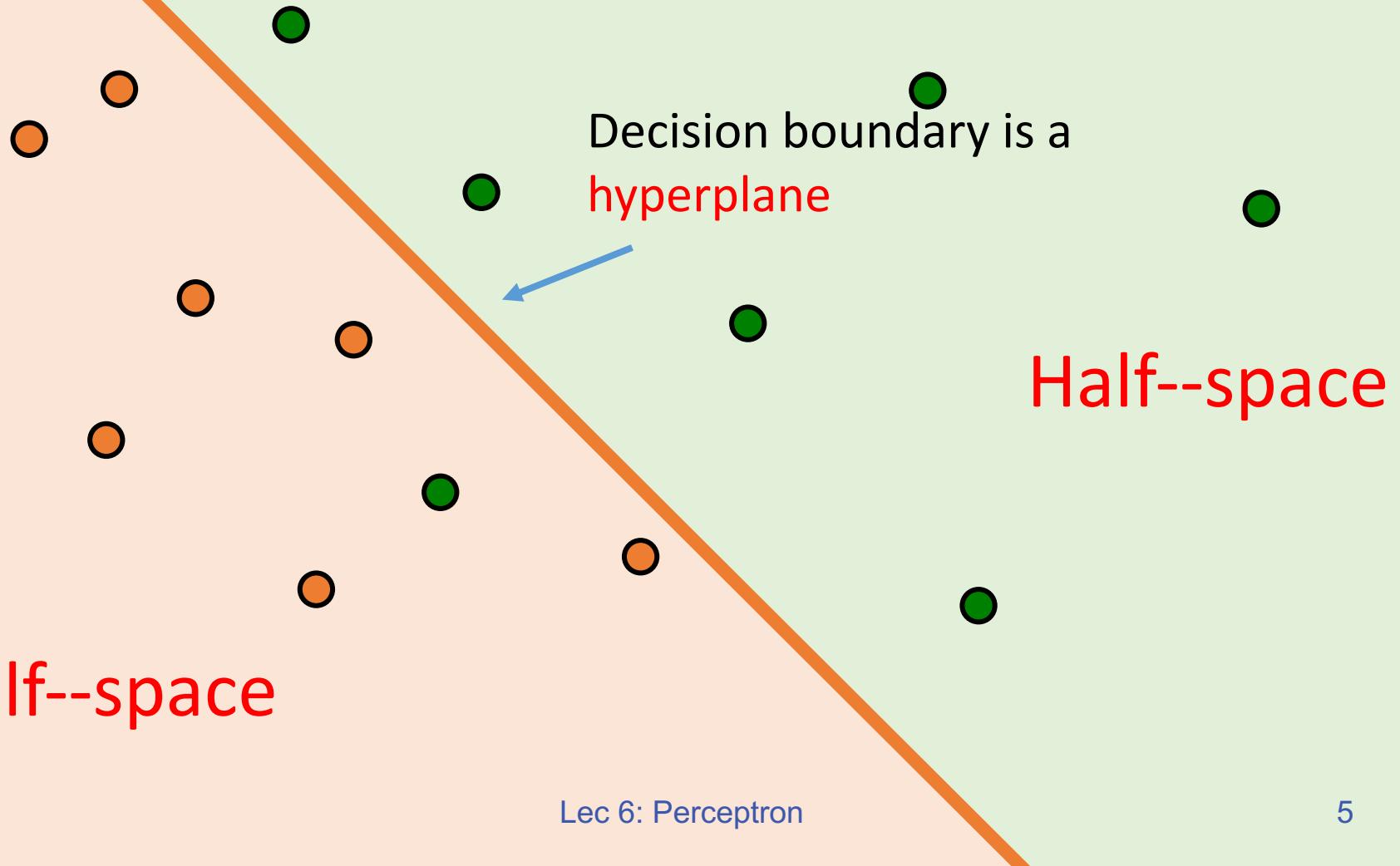
What you will Learn Today

- ❖ Linear model
 - ❖ Basic linear algebra & linear classifier
 - ❖ Trick to remove bias term b in $w^T x + b = 0$
- ❖ Perceptron Algorithm
 - ❖ Perceptron Update
 - ❖ Why it works
 - ❖ Convergence theorem – mistake bound

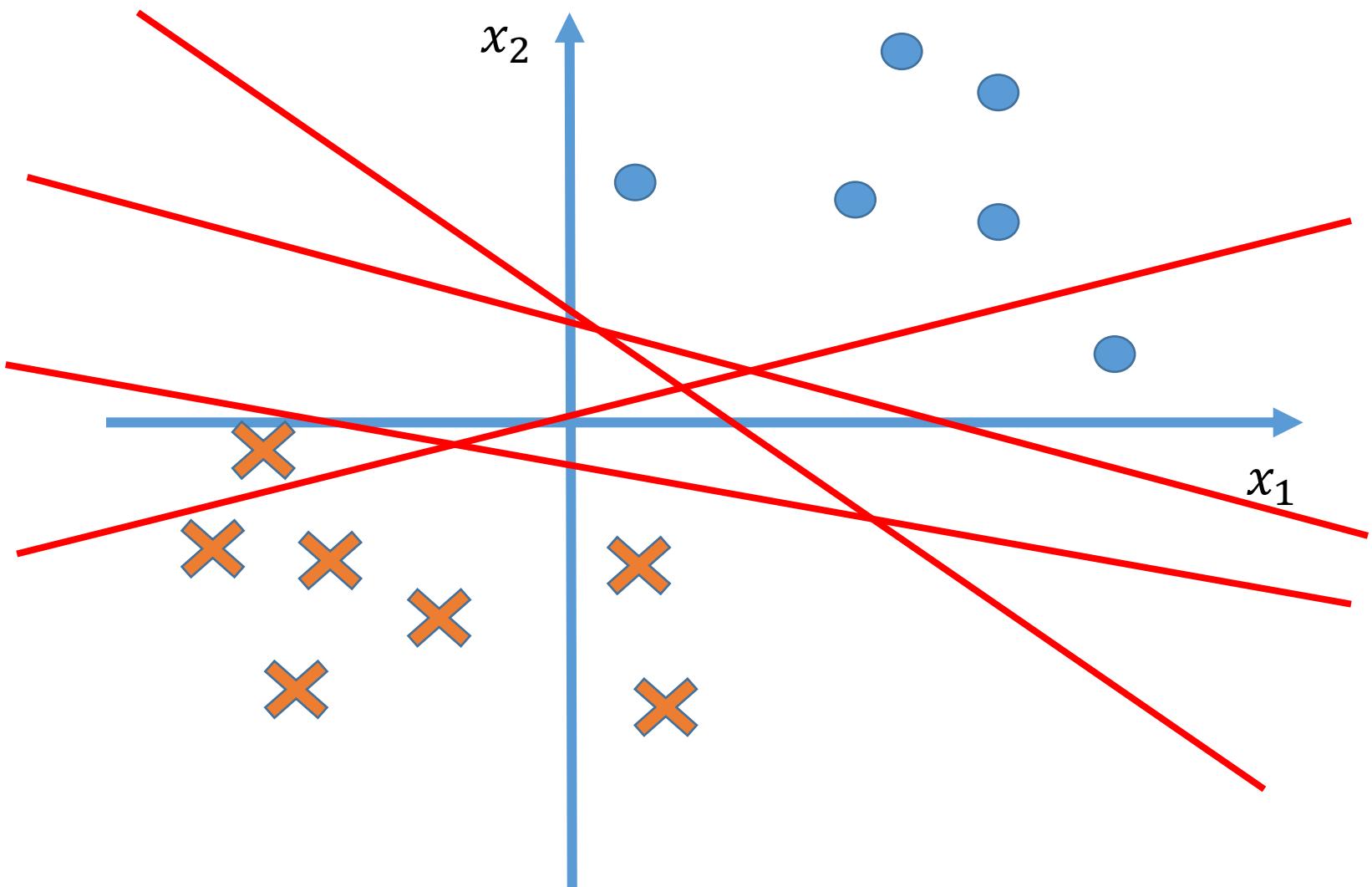
Training data



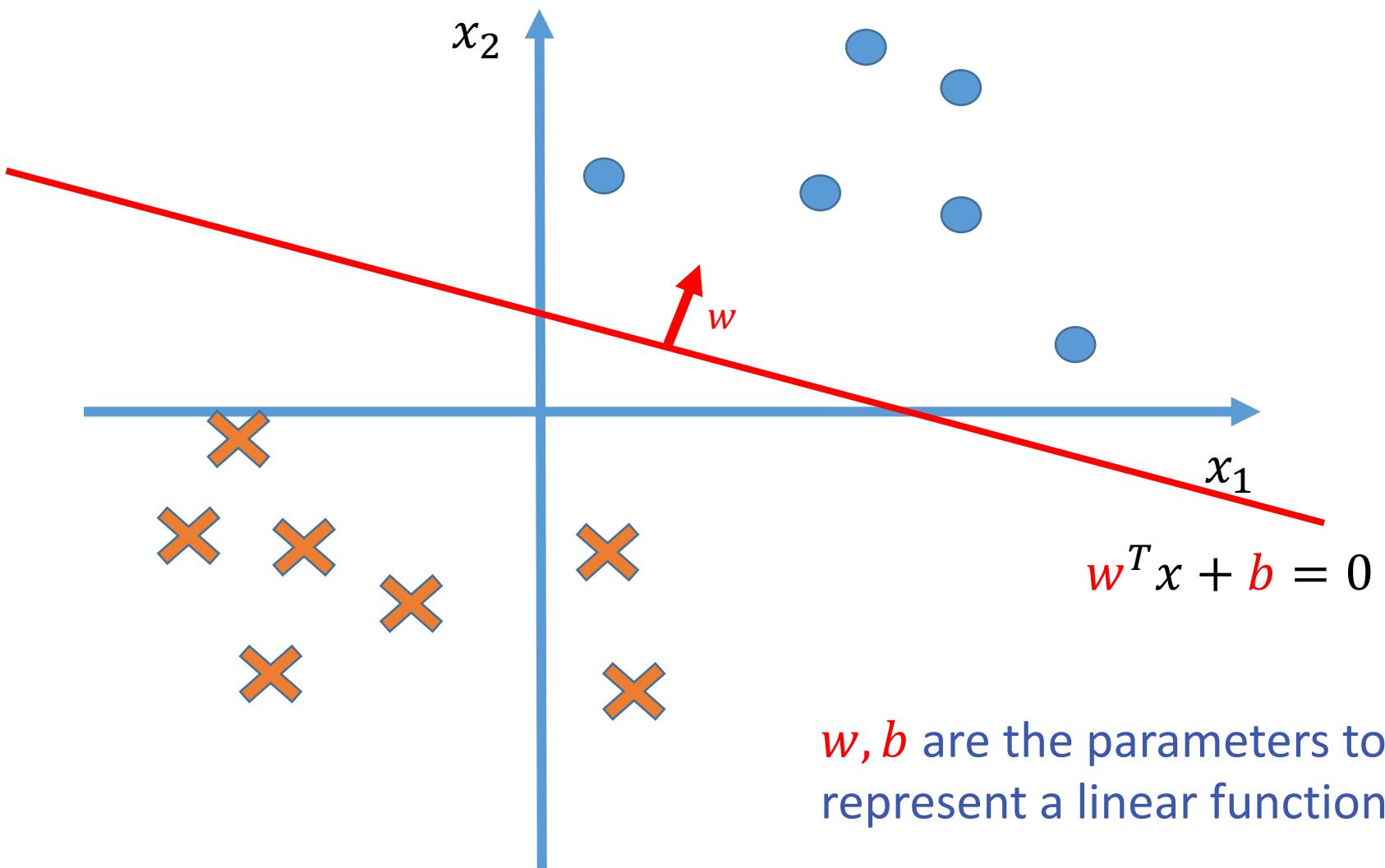
Hyperplane Separates the Space



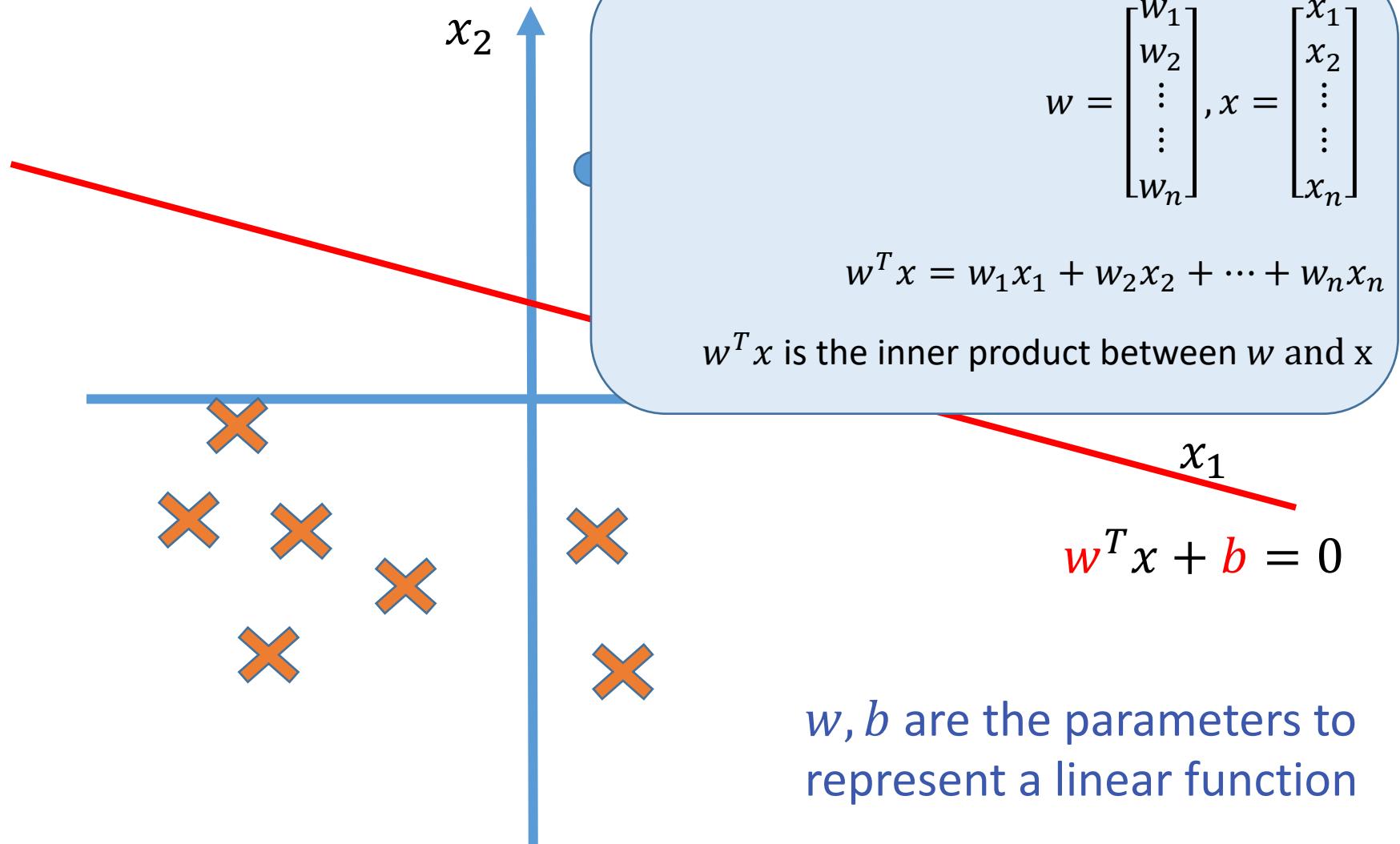
Hypothesis space: linear models



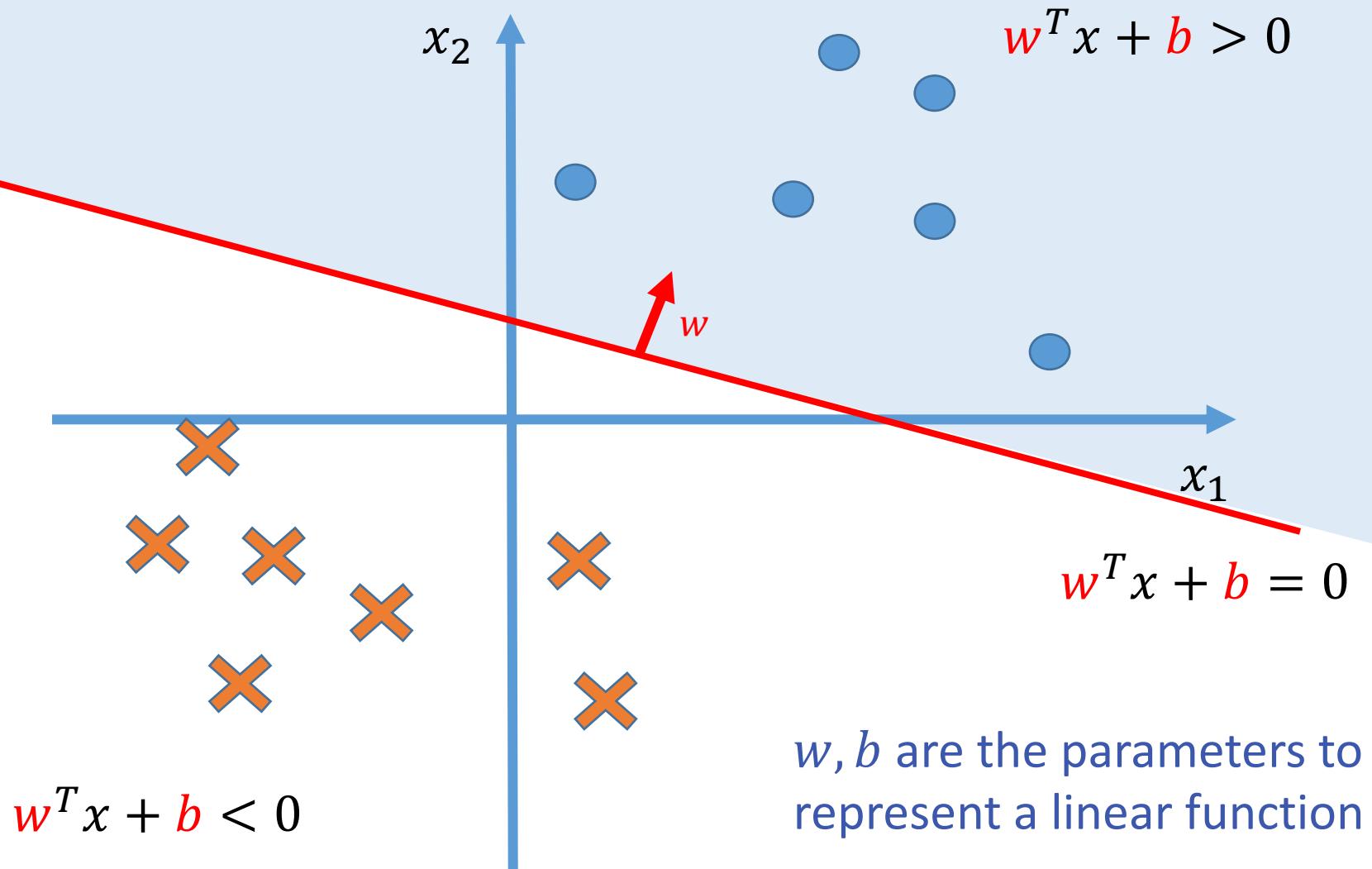
Hypothesis space: linear model



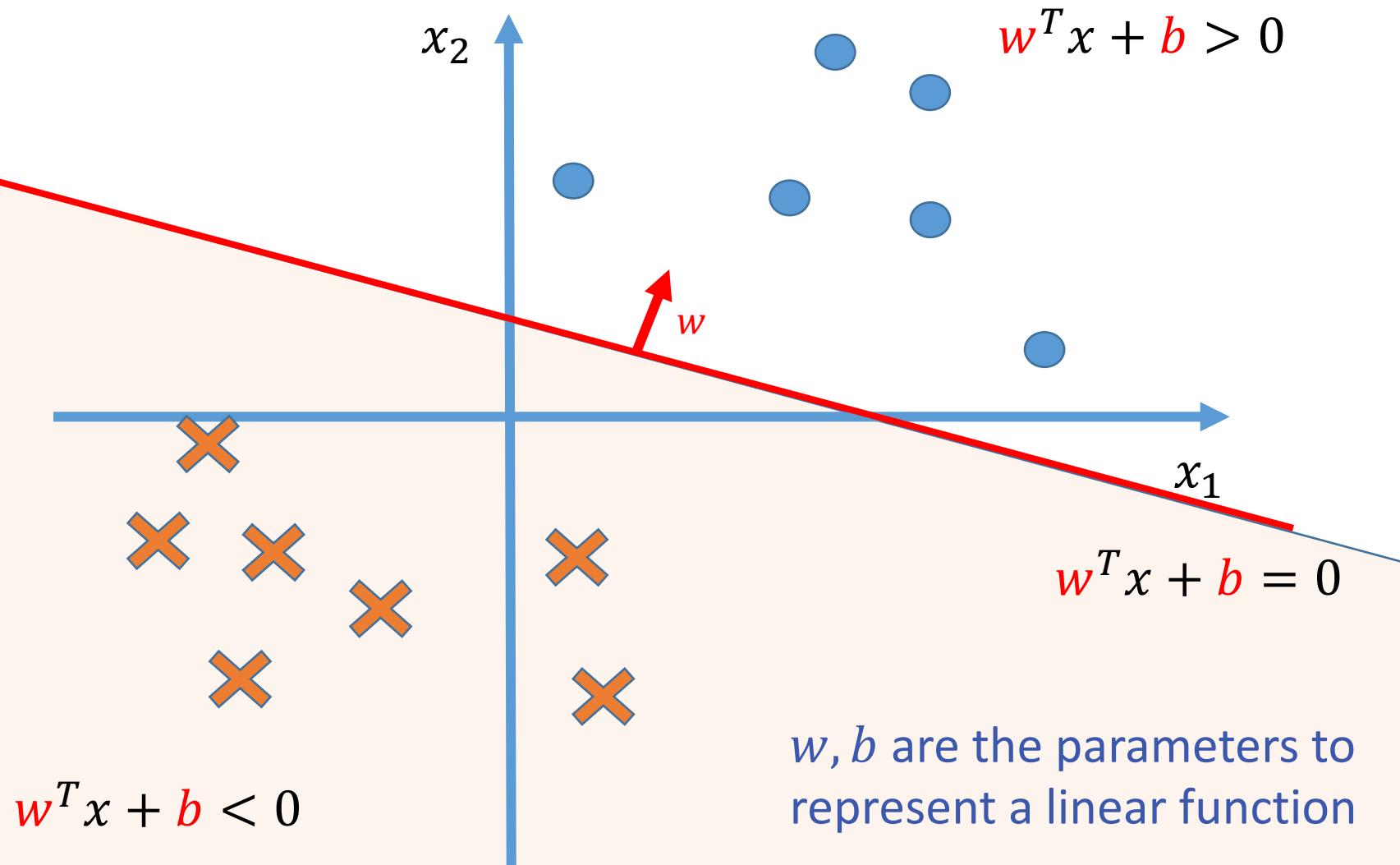
Hypothesis space: linear model



Hypothesis space: linear model

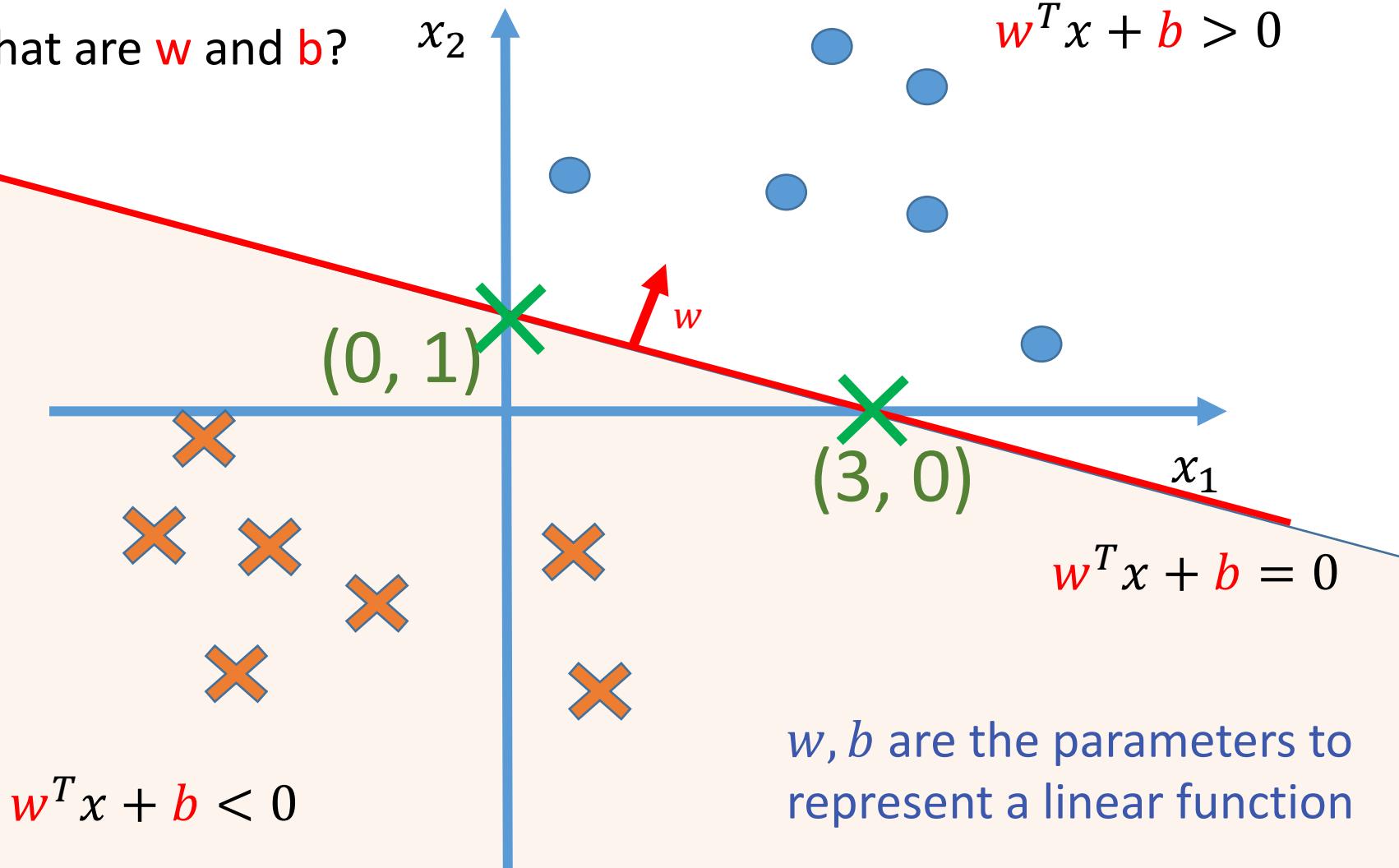


Hypothesis space: linear model



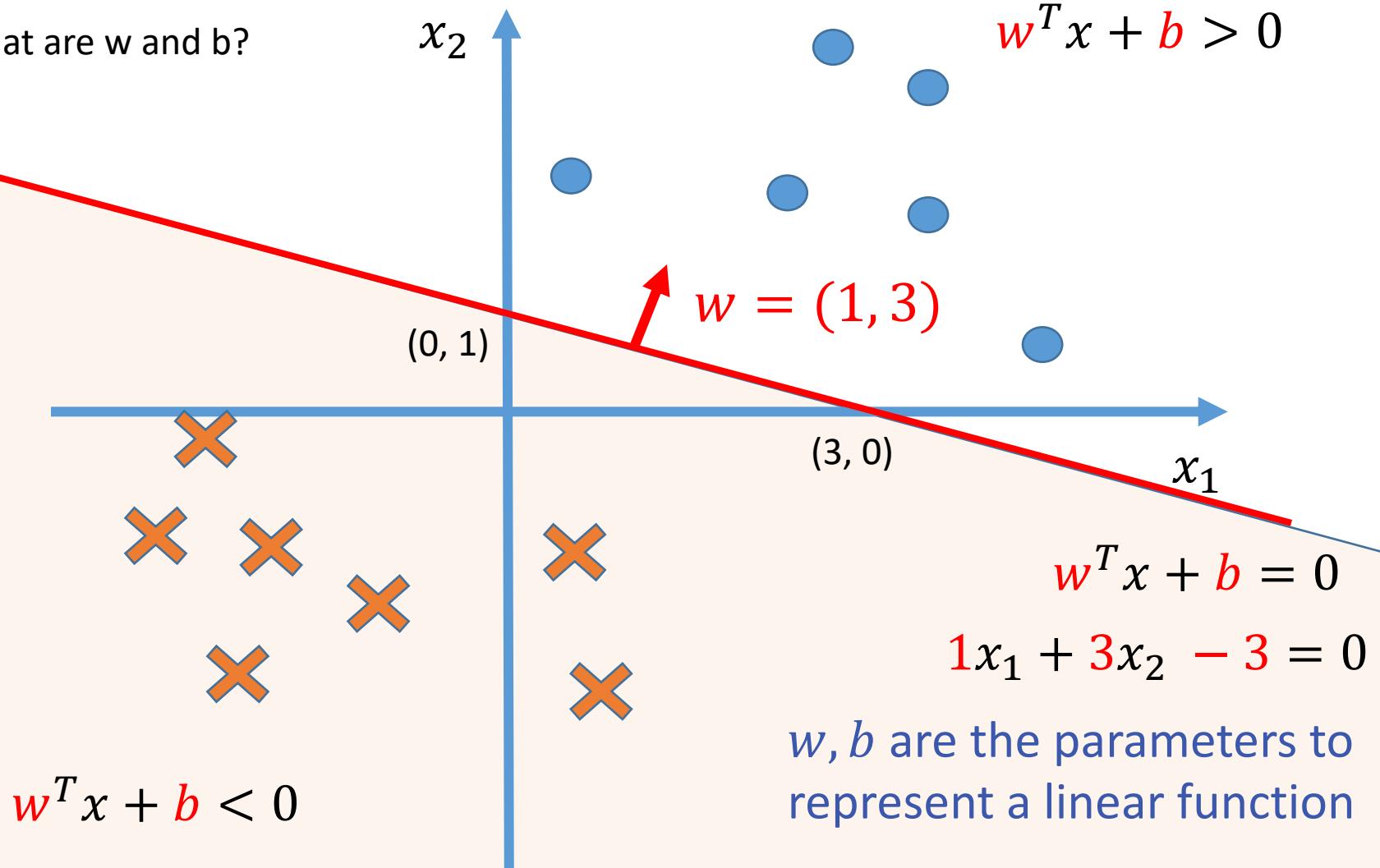
Example

What are w and b ?



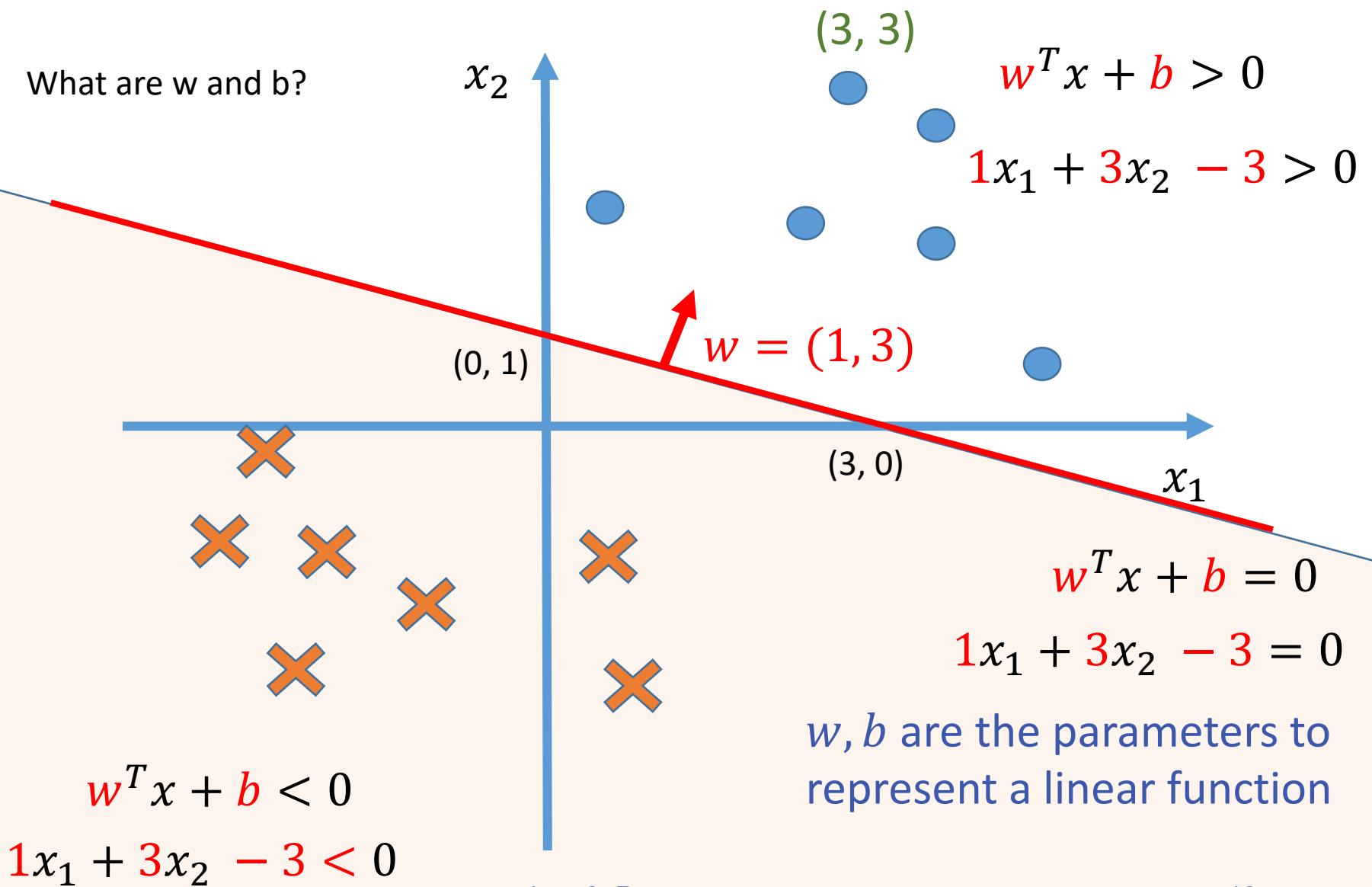
Example

What are w and b ?



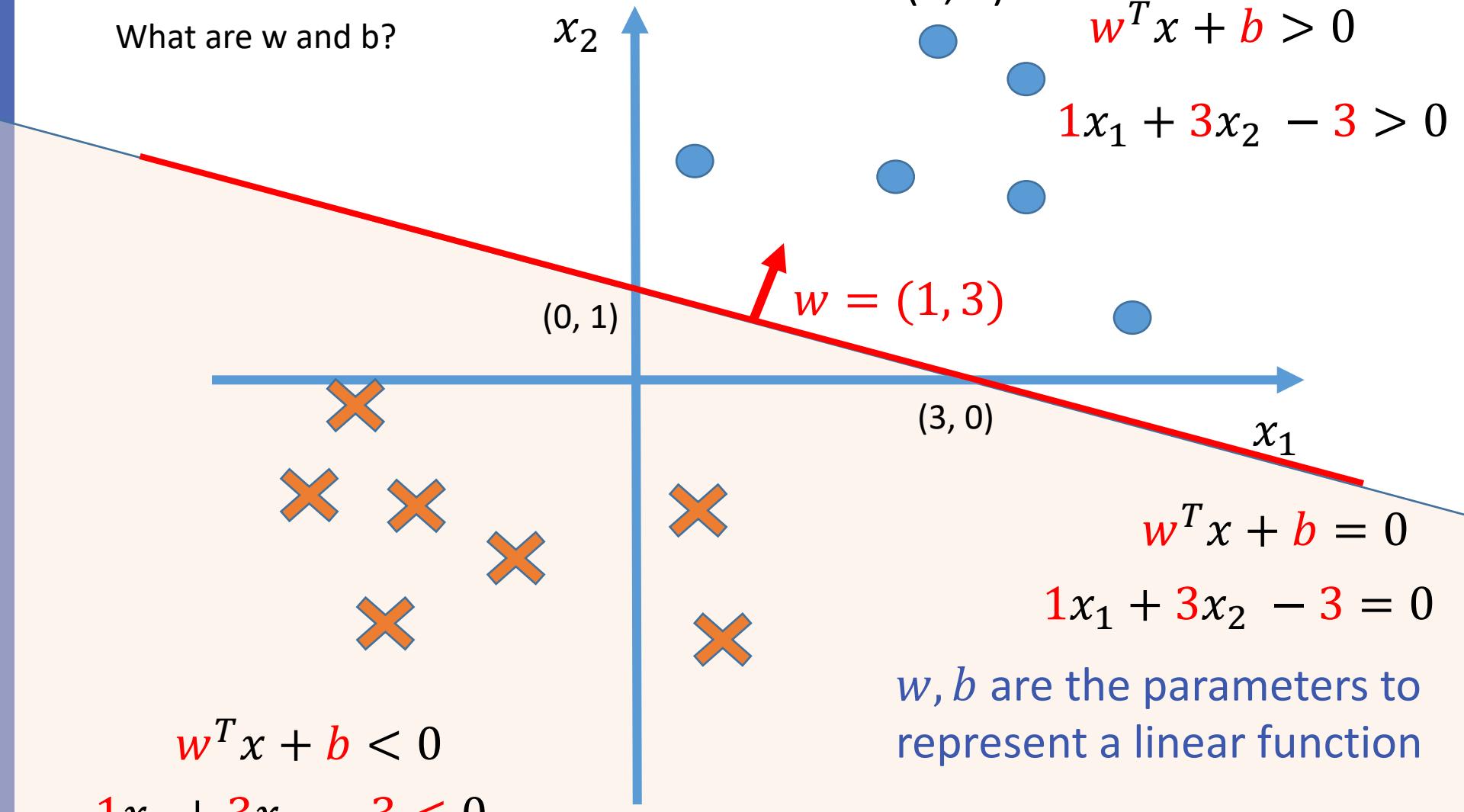
Example

What are w and b ?

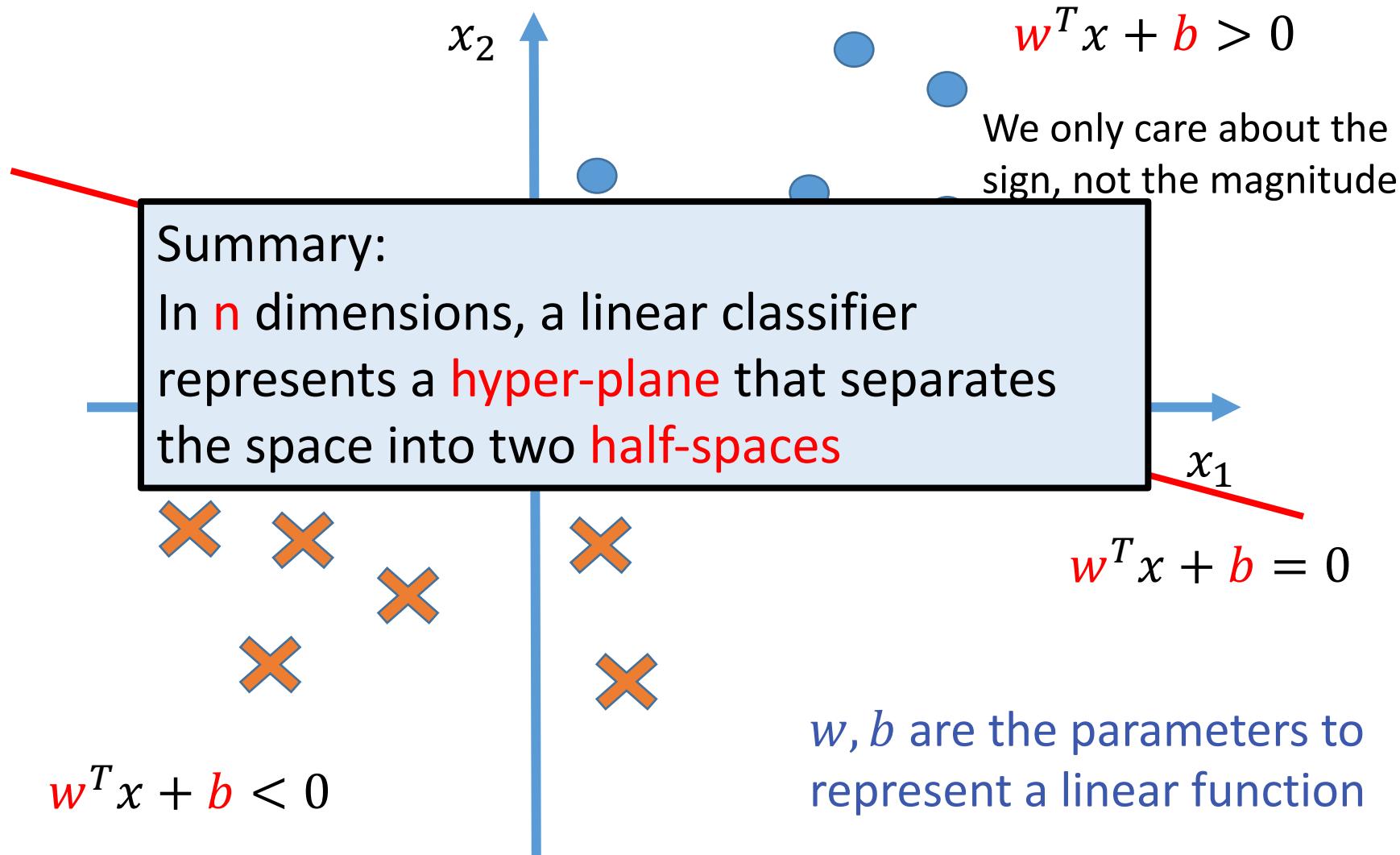


Example

What are w and b ?



Hypothesis space: linear model



Linear Models for Binary Classification

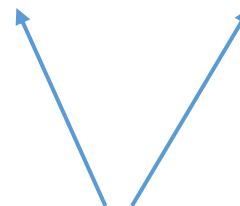
- Given training set $\mathcal{D} = \{(x, y)\}, x \in \mathbb{R}^d, y \in \{-1, +1\}$, we use them to learn a hypothesis function $h \in H$

$$H = \{ h \mid h(x) = \text{sgn}(w^T x + b) \}$$

such that $y = h(x)$

$$\text{sgn}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Note: when $z=0$ we can either assign $\text{sgn}(z) = 1$ or -1



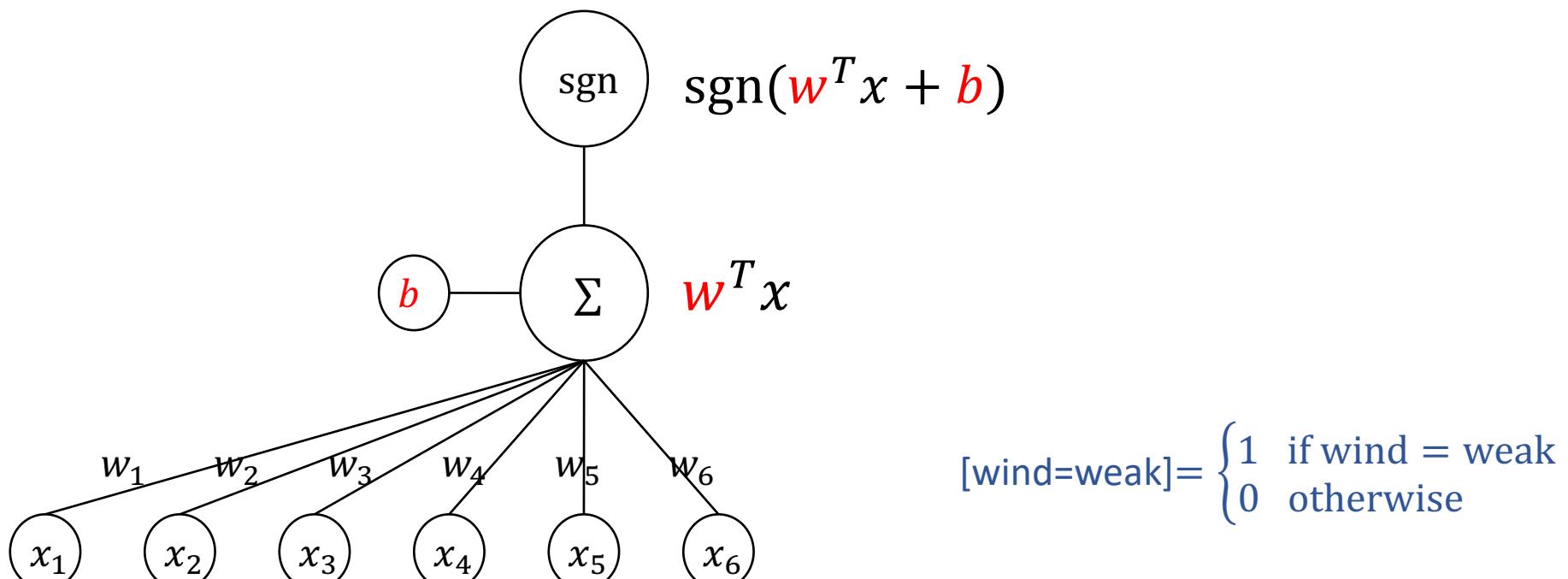
model parameters
(learnable parameters)

Learn = train = find the best parameters w, b

Footnote: For some algorithms it is mathematically easier to represent False as -1 , and at other times, as 0 . For the Perceptron algorithm, treat -1 as false and $+1$ as true.

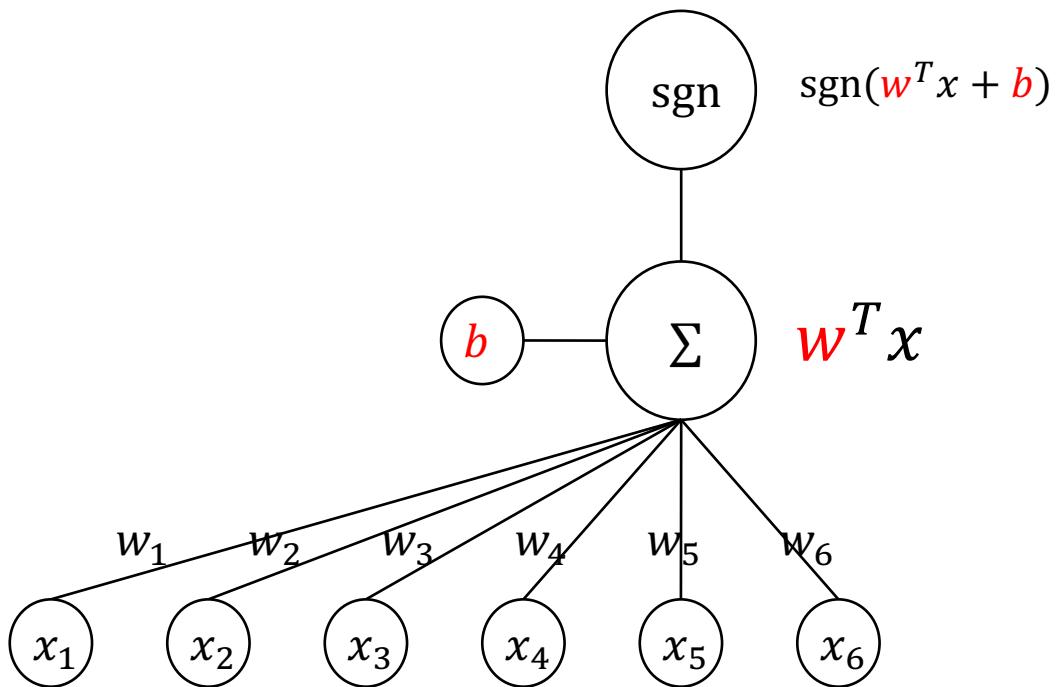
Linear Classifiers

- ❖ *Linear classifiers* classify an example \mathbf{x} using the following classification rule



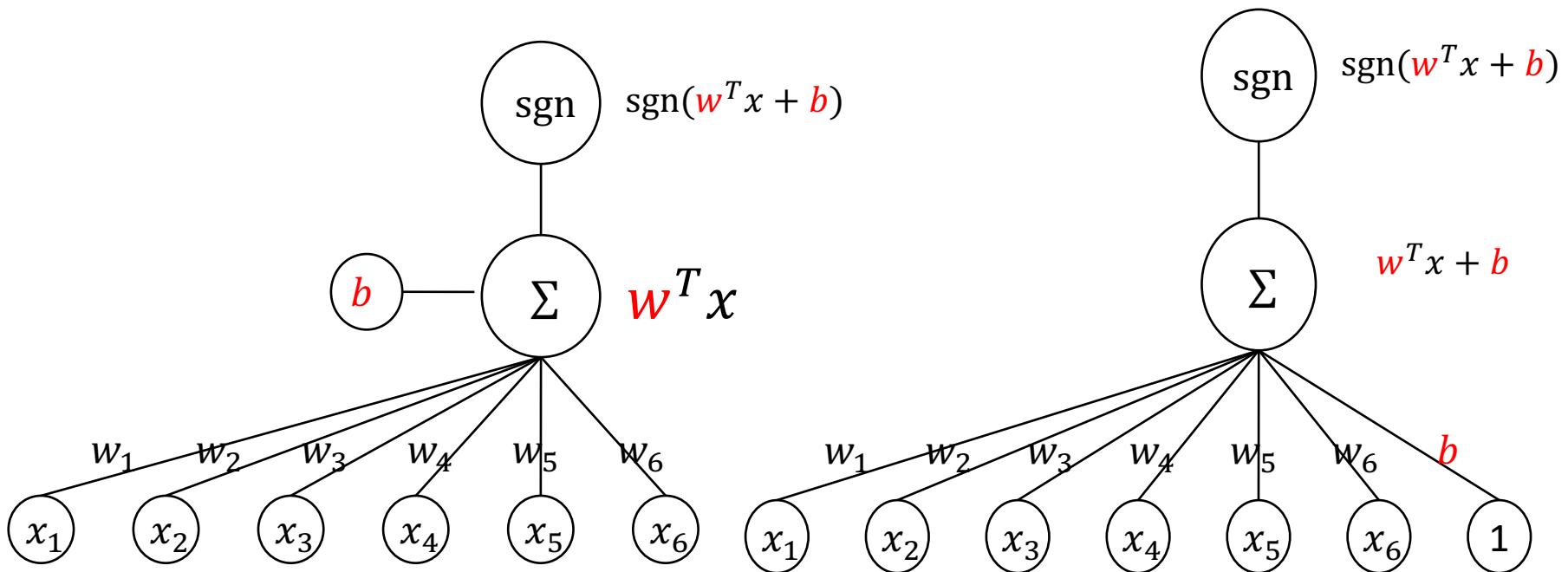
E.g., $3 * [\text{wind=weak}] + 6 * [\text{outlook=sunny}] - 2 * [\text{temperature=high}] + \dots - 2 > 0$

Linear Classifiers



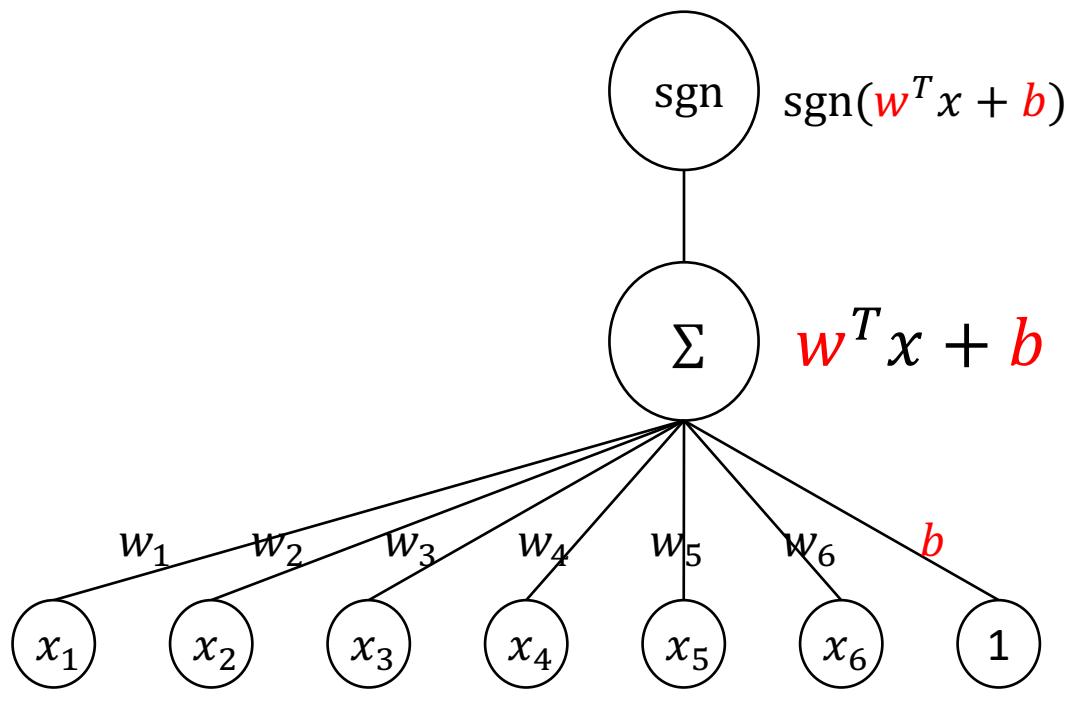
E.g., $3 * [\text{wind=weak}] + 6 * [\text{outlook=sunny}] - 2 * [\text{temperature=high}] + \dots - 2 > 0$

Linear Classifiers



E.g., $3 * [\text{wind=weak}] + 6 * [\text{outlook=sunny}] - 2 * [\text{temperature=high}] + \dots - 2 > 0$

A simple trick to remove the bias term b

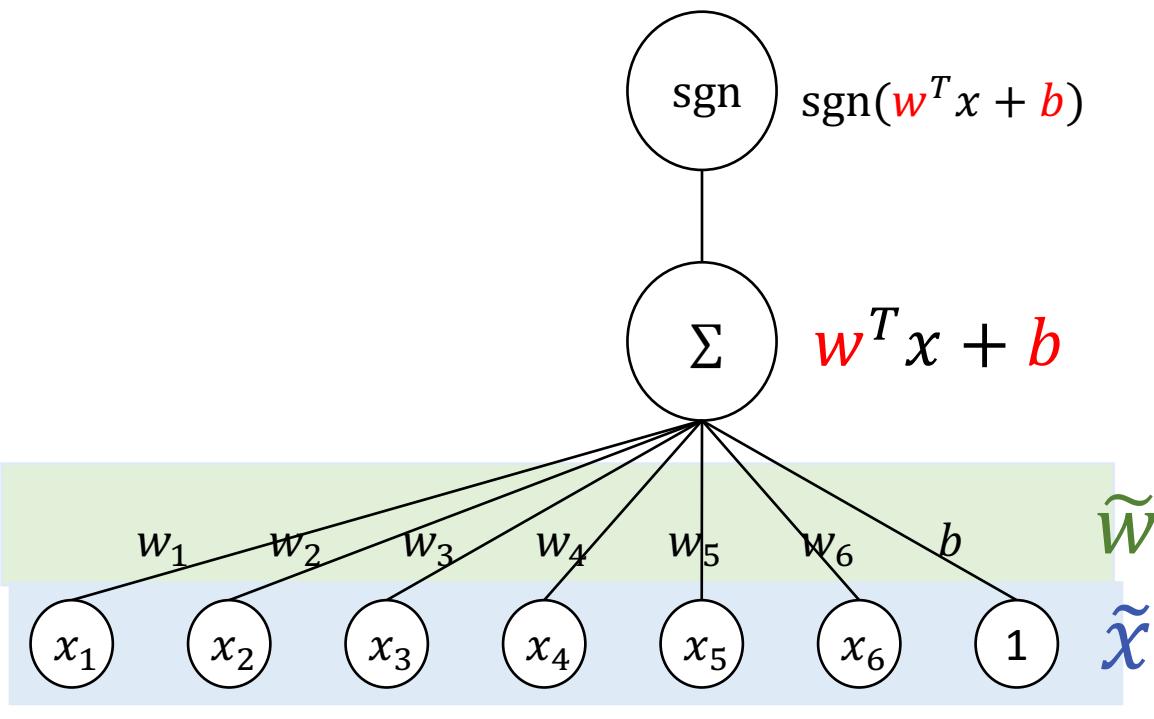


$$\begin{aligned} w^T x + b \\ = [w^T \ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\ = \tilde{w} \cdot \tilde{x} \end{aligned}$$

$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

A simple trick to remove the bias term b



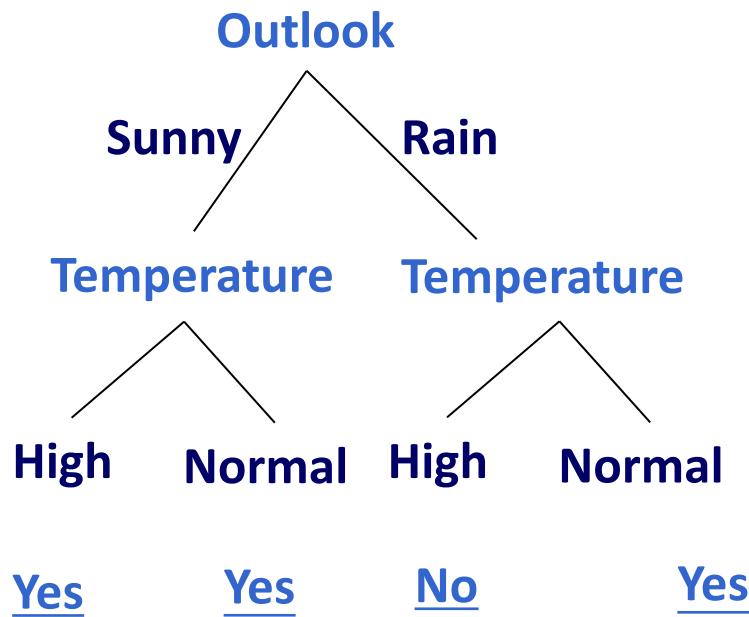
$$\begin{aligned} w^T x + b \\ = [w^T \ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\ = \tilde{w} \cdot \tilde{x} \end{aligned}$$

$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

Exercise

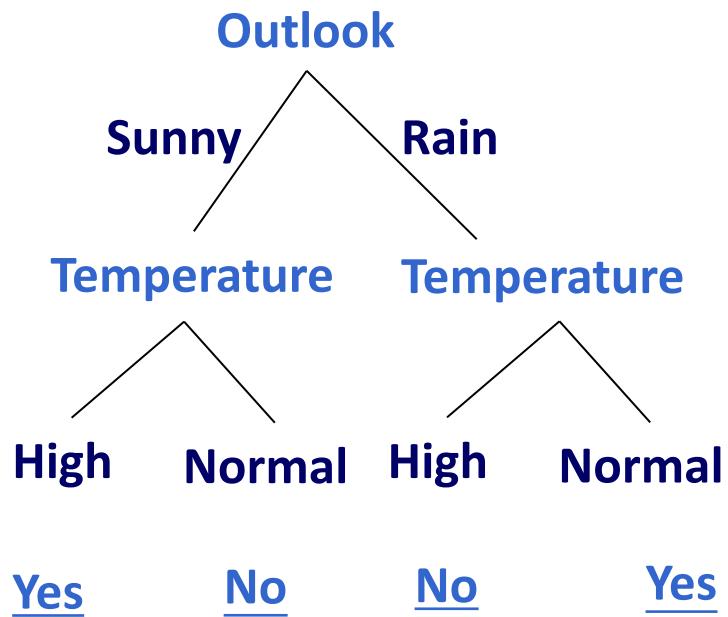
Find a linear model that is equivalent to the decision tree on the left



$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} [\text{outlook} = \text{sunny}] \\ [\text{outlook} = \text{rain}] \\ [\text{temp} = \text{high}] \\ [\text{temp} = \text{normal}] \\ 1 \end{bmatrix}$$

Exercise

Find a linear model that is equivalent to the decision tree on the left



$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} [\text{outlook} = \text{sunny}] \\ [\text{outlook} = \text{rain}] \\ [\text{temp} = \text{high}] \\ [\text{temp} = \text{normal}] \\ 1 \end{bmatrix}$$

Learning a Linear Classifier

Learn = train = find the **best parameters w, b**

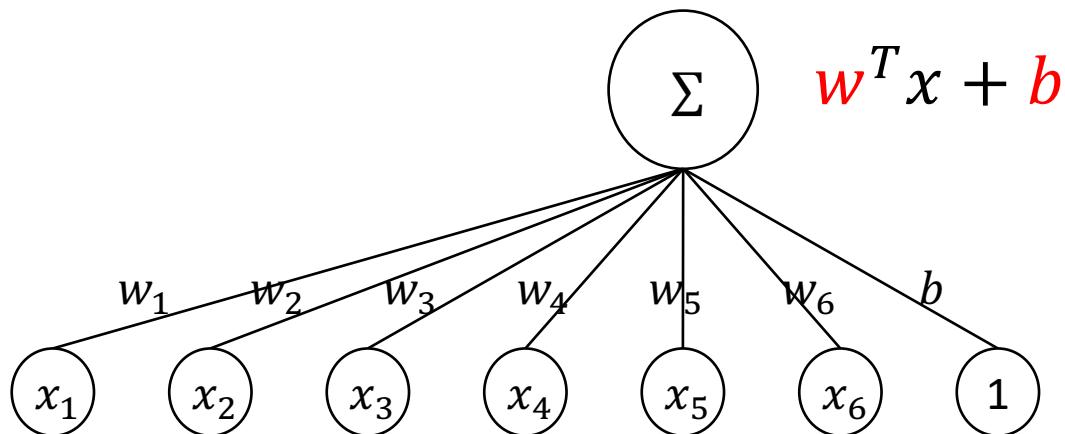
- ❖ There are several algorithms/models
 - ❖ Perceptron
 - ❖ Logistic Regression
 - ❖ (Linear) Support Vector Machines
 - ❖ Naïve Bayes
 - ❖ ...
- ❖ Different methods define **best** in a different way

Linear Regression

- ❖ *Linear regression* maps an example \mathbf{x} into a real value y
- ❖ Given training set $\mathcal{D} = \{(\mathbf{x}, y)\}, \mathbf{x} \in R^d, y \in R$, we use them to learn a hypothesis function $h \in H$

$$H = \{ h \mid h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \}$$

such that $y = h(\mathbf{x})$



Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN¹

F. ROSENBLATT

Cornell Aeronautical Laboratory

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

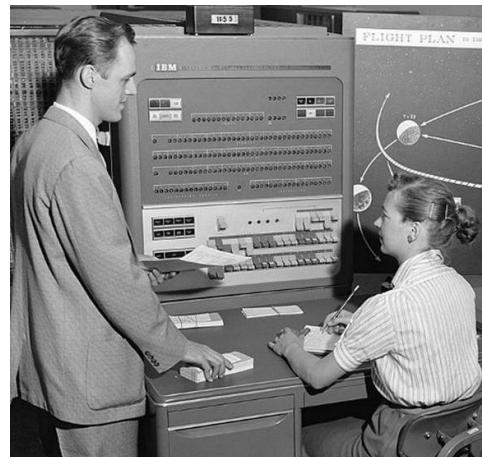
WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44





Note: The application scenario discussed in the video is not ideal.

The Perceptron algorithm

- ❖ The goal is to find a separating hyperplane
- ❖ An **online** algorithm
 - ❖ Processes one example at a time
- ❖ Converges if data is separable
 - mistake bound

What you will learn about Perceptron

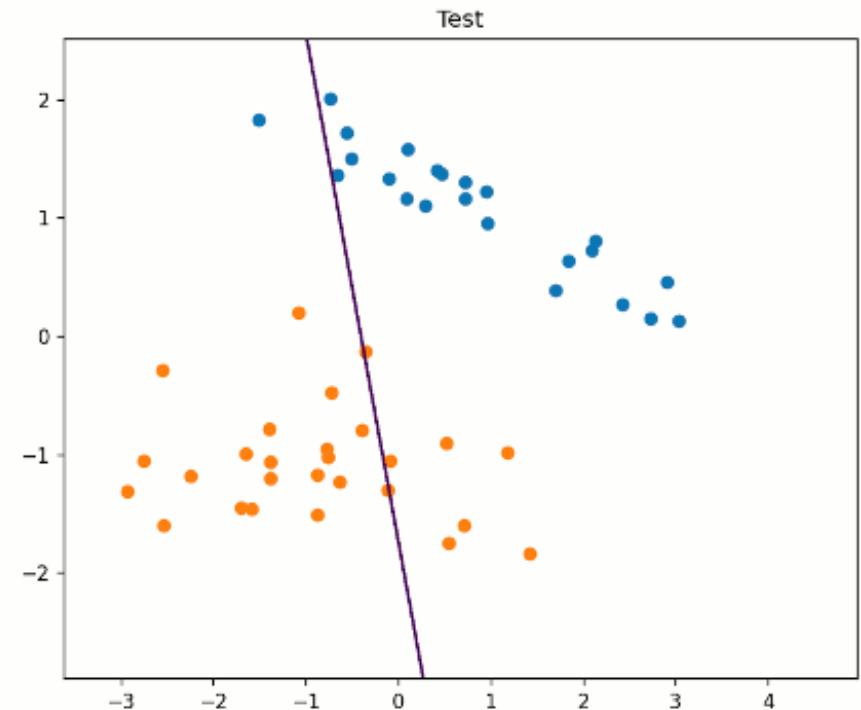
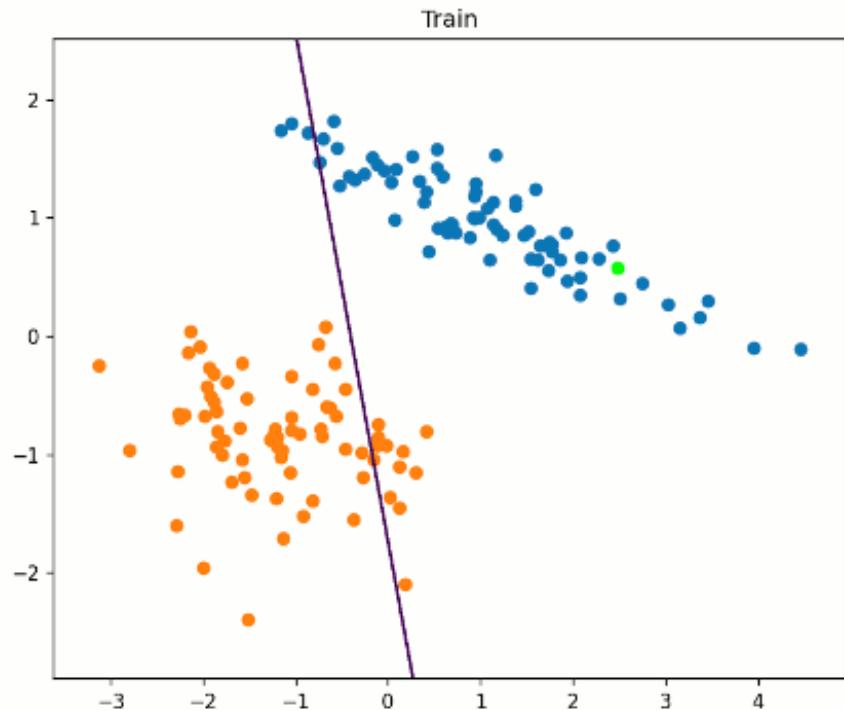
- ❖ Perceptron Algorithm
- ❖ The intuition behind the algorithm
- ❖ Convergence of Perceptron Algorithm
- ❖ Mistake bound

mistake
+
correction
=

learning

Perceptron in Action

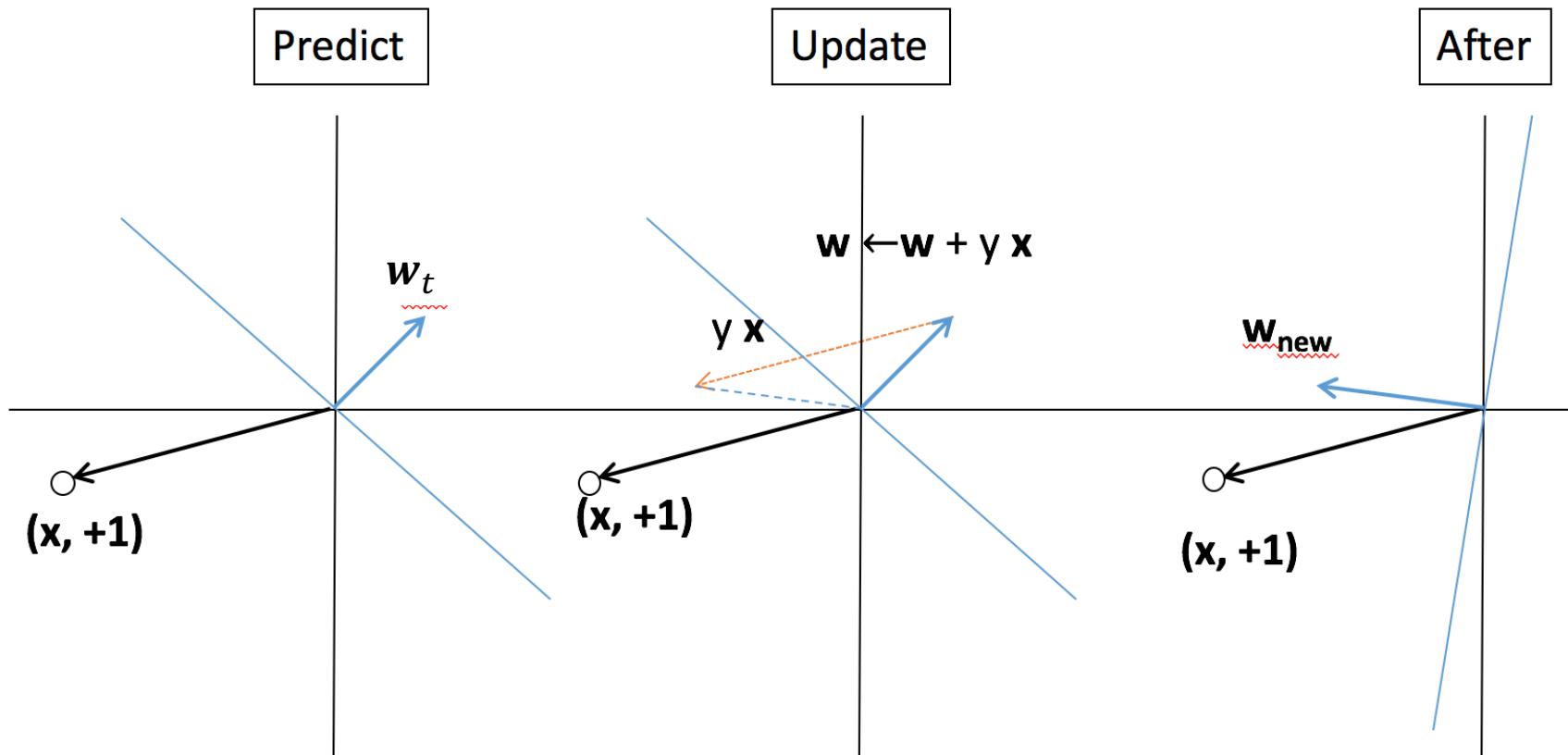
Iteration: 1/2; Point: 1/150



<https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1>

Perceptron

- ❖ Learning by making mistakes

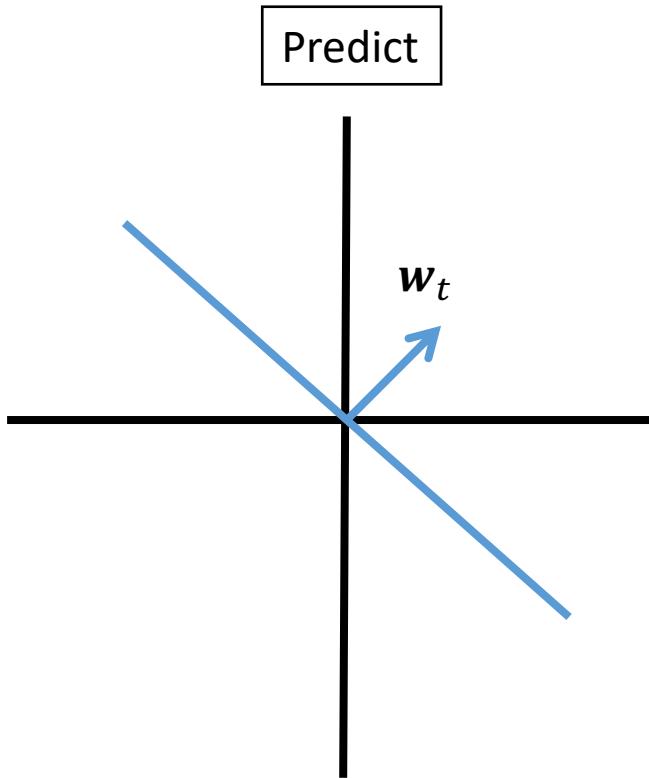


The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

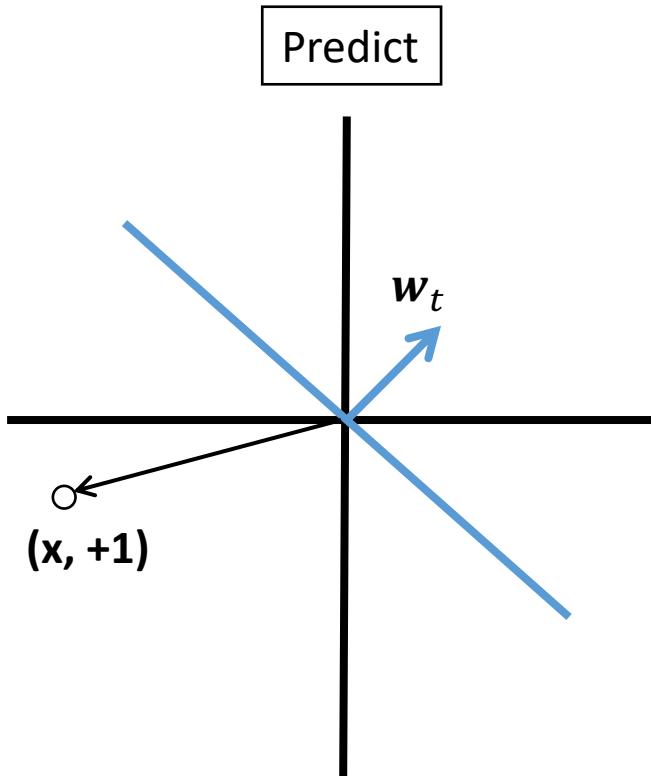
Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Geometry of the perceptron update



Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

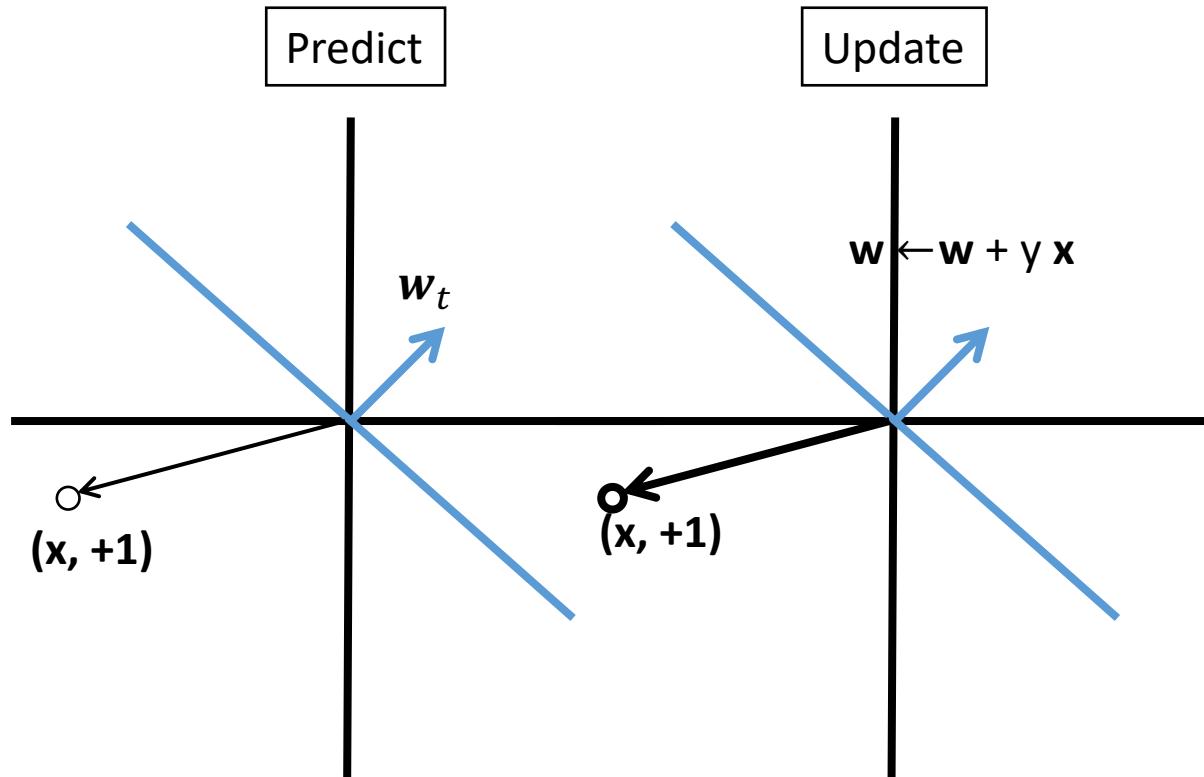
Geometry of the perceptron update



Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Geometry of the

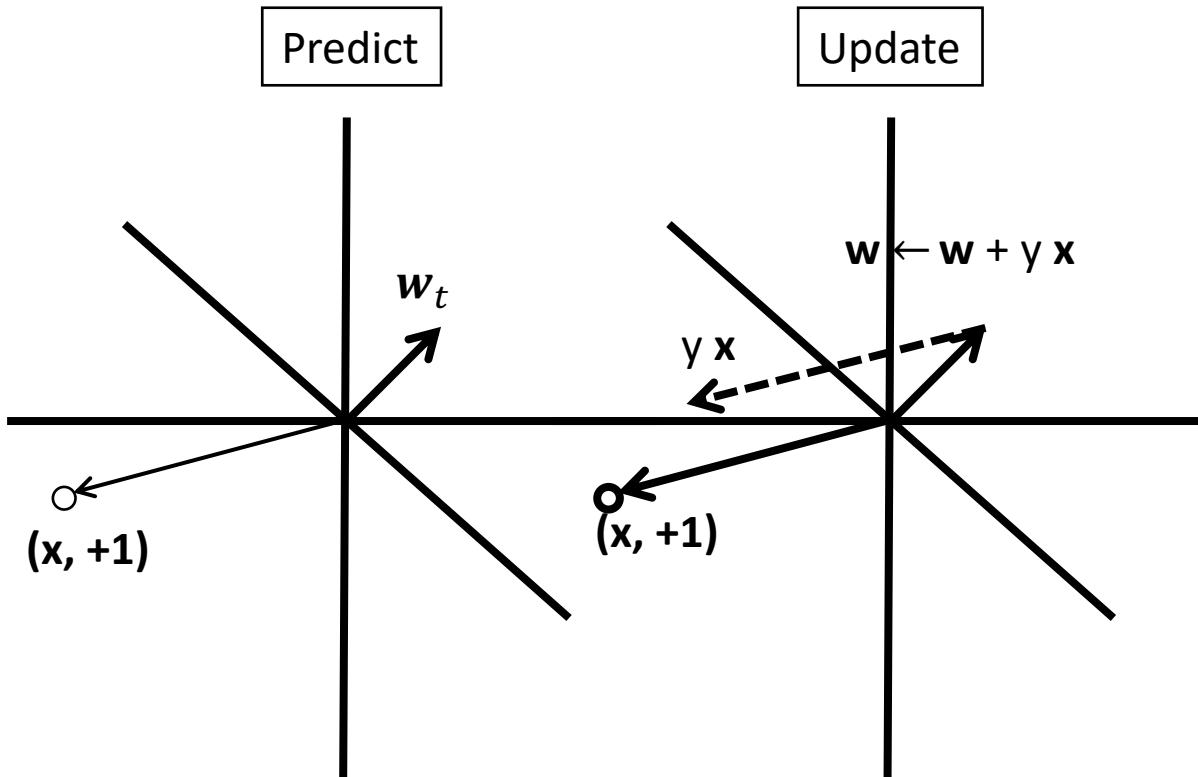
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

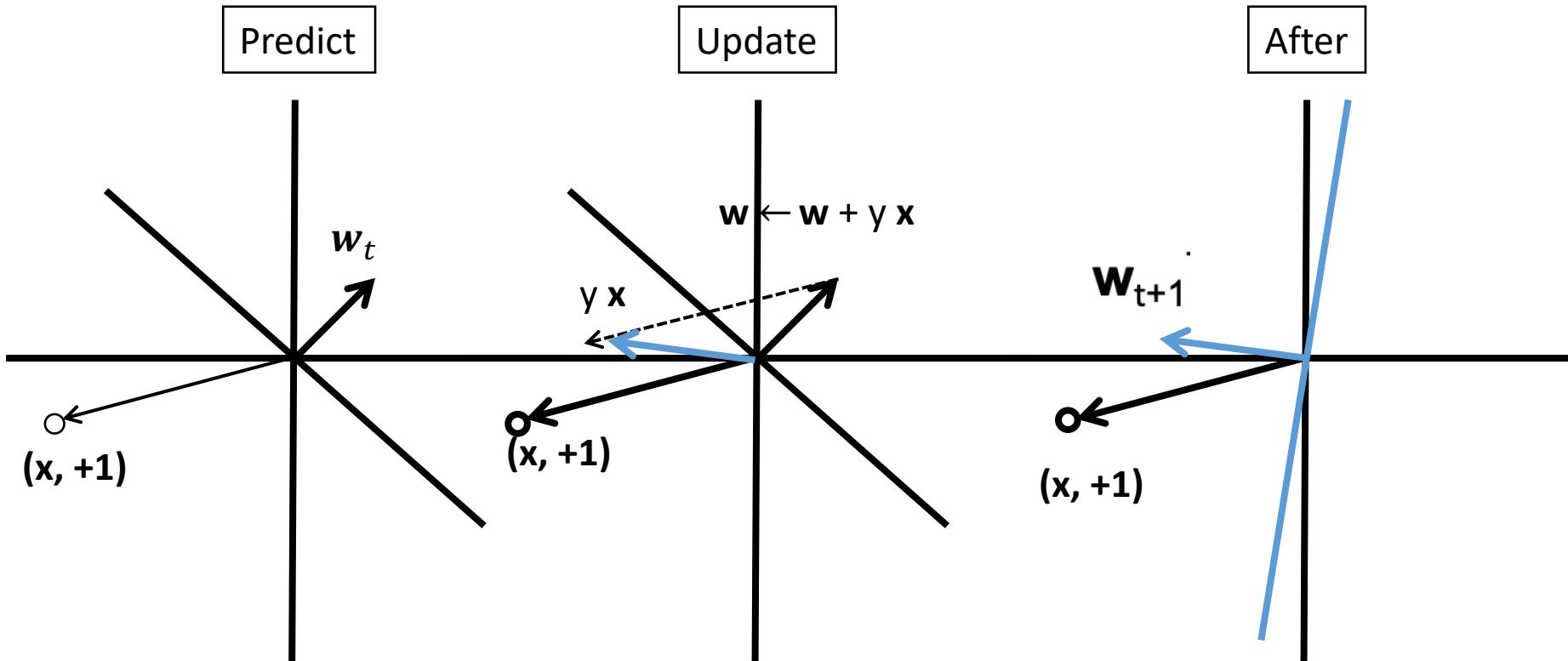
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

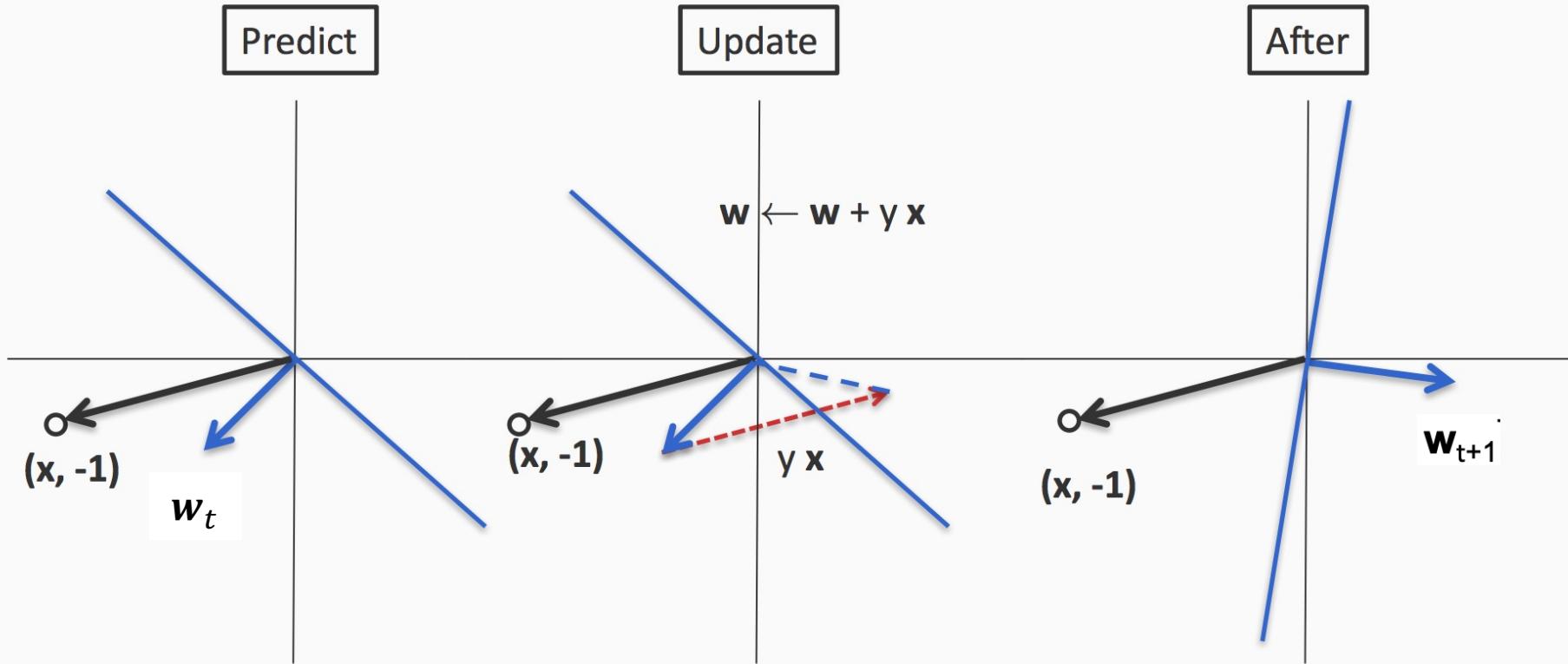
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **negative** example

Intuition behind the update

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Suppose we have made a mistake on a positive example

That is, $y = +1$ and $\mathbf{w}_t^T \mathbf{x} \leq 0$

Call the new weight vector $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$

The new dot product will be

$$\mathbf{w}_{t+1}^T \mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T \mathbf{x} = \mathbf{w}_t^T \mathbf{x} + \mathbf{x}^T \mathbf{x}$$

For a positive example, the Perceptron update will increase the score assigned to the same input

Similar reasoning for negative examples

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
 2. For (\mathbf{x}, y) in \mathcal{D} :
 3. $\hat{y} = \text{sgn}(\mathbf{w}^\top \mathbf{x})$
 4. if $\hat{y} \neq y$, $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
 - 5.
 6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm (batch)

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
 For (x, y) in \mathcal{D} :
 if $y(w^T x) \leq 0$
 $w \leftarrow w + \eta yx$
3. Return w



η is a hyper-parameter to the algorithm

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T x^{\text{test}})$

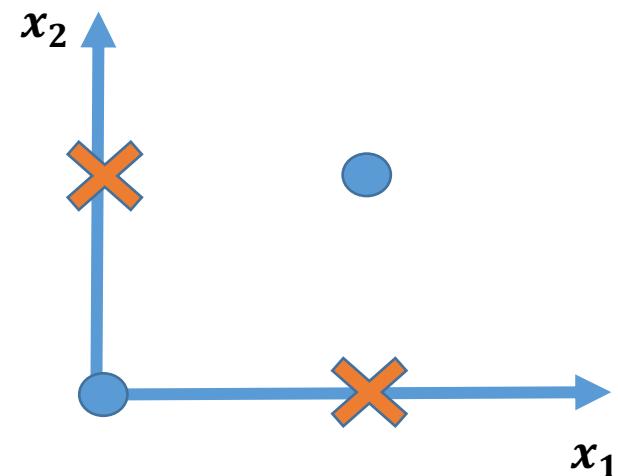
Check point: What you need to know

- ❖ The Perceptron algorithm
- ❖ The geometry of the update

Perceptron Learnability

- ❖ Perceptron cannot learn what it cannot represent
 - ❖ Only linearly separable functions
 - Minsky and Papert (1969)
 - ❖ E.g., Parity functions can't be learned (XOR)

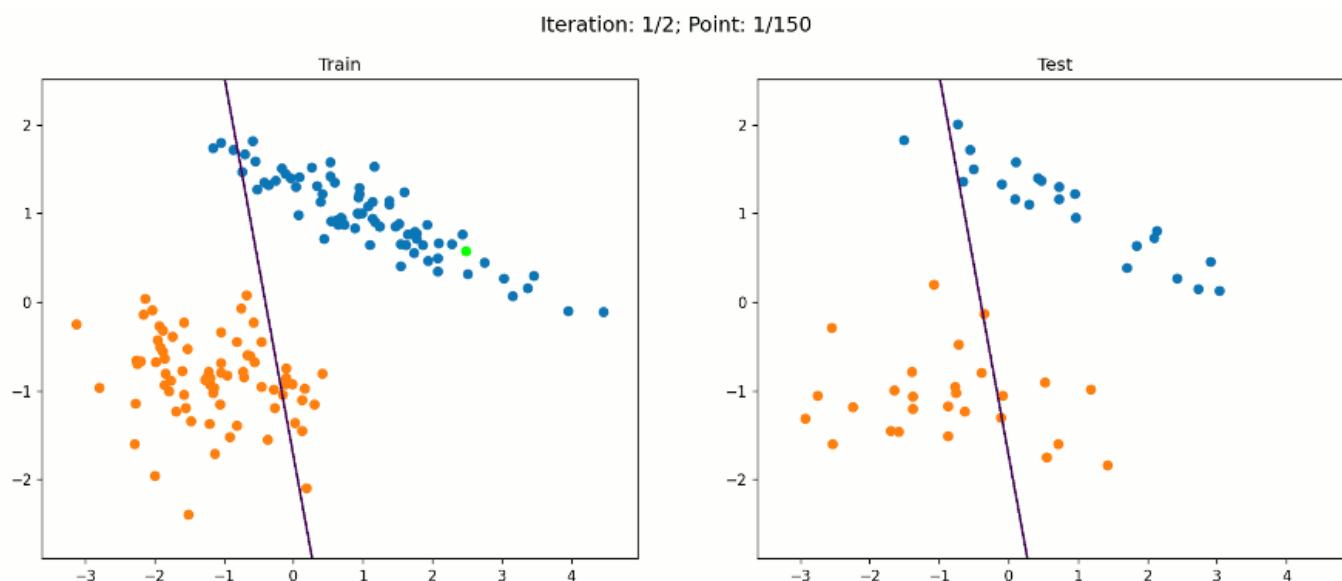
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



Convergence

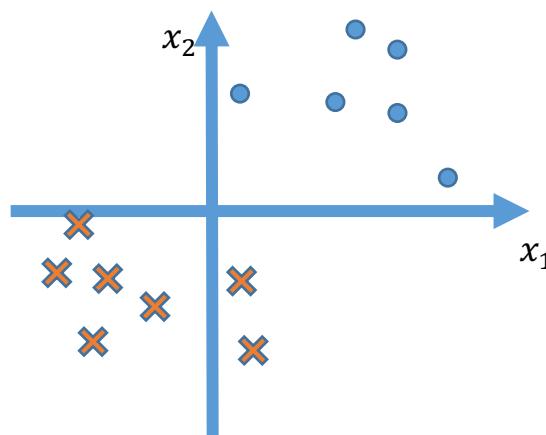
Convergence theorem

- ❖ If there exists a model that is consistent with the data (i.e. the data is **linearly separable**), the perceptron algorithm will converge.

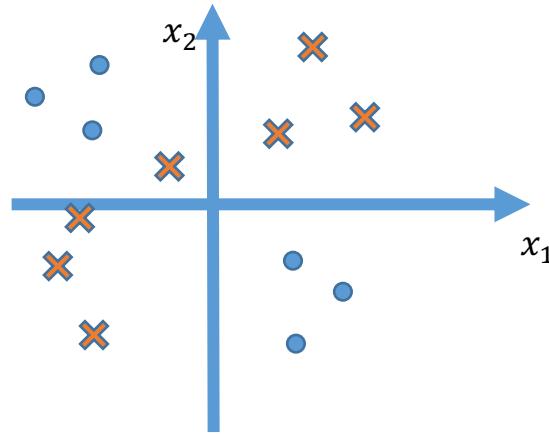


Convergence theorem

- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is **linearly separable**), the perceptron algorithm will converge.
- ❖ Note: this is the condition of the data we may not know what the hyperplane is



Linearly separable data

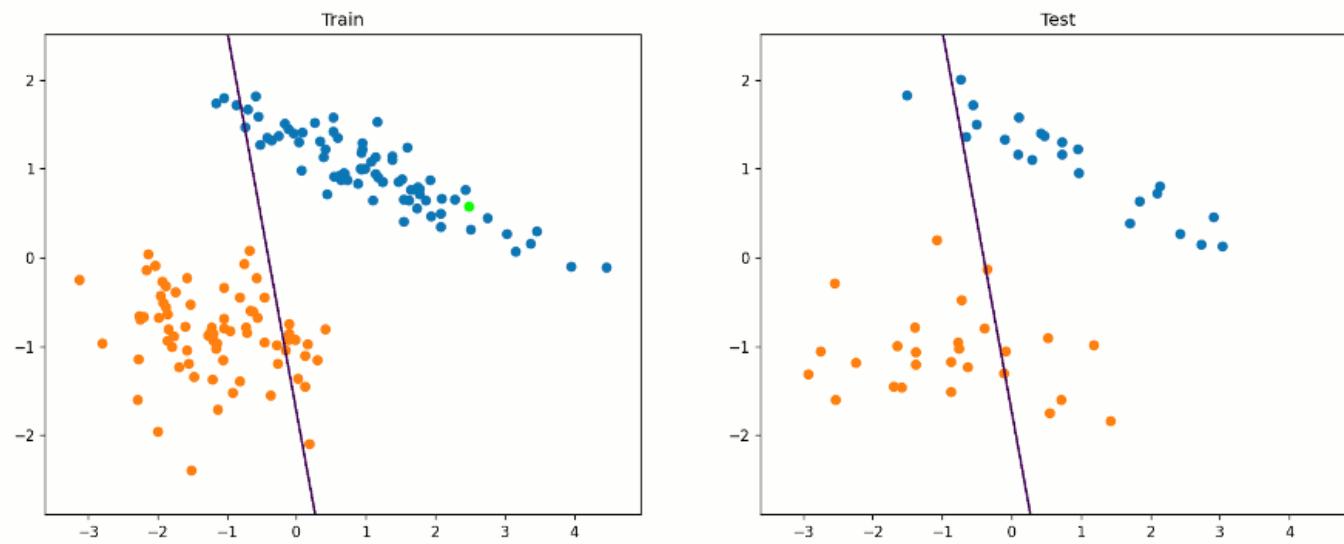


Linearly non-separable data

Convergence theorem

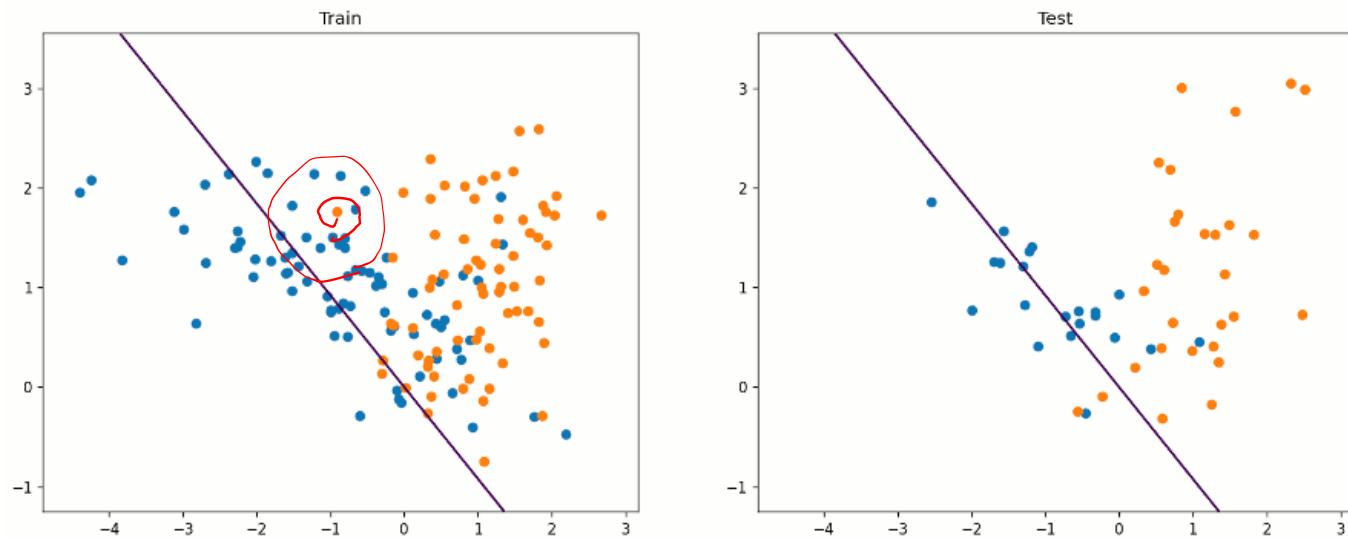
- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge
 - ❖ The update stops after making a finite number of mistakes.
 - ❖ The convergence rate depends on the difficulty of the problem (explain later)
- ❖ If the training data **is not linearly** separable, then the learning algorithm will eventually repeat the same set of weights and **enter an infinite loop**

Iteration: 1/2; Point: 1/150



Linearly Separable

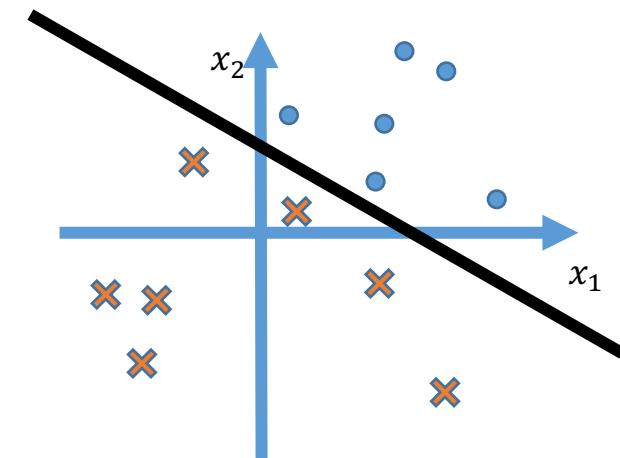
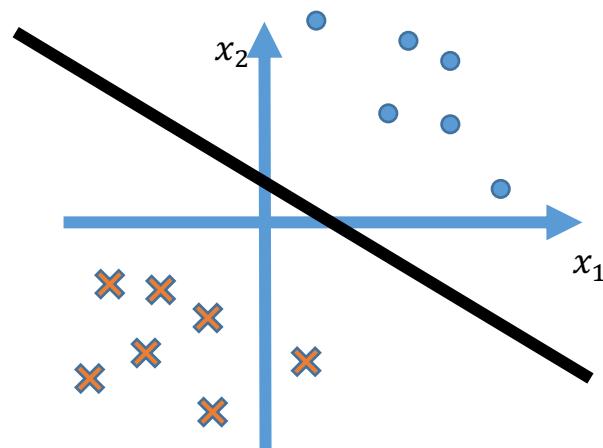
Iteration: 1/100



Not Linearly Separable

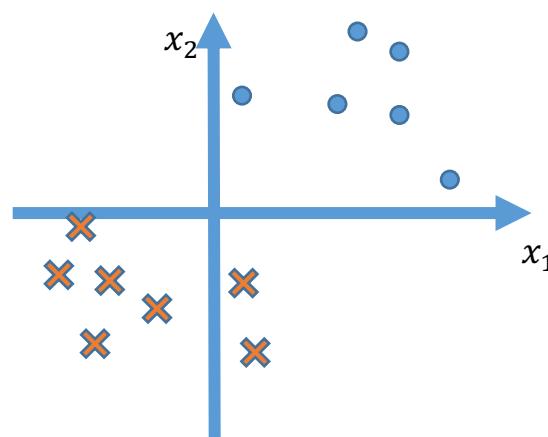
Convergence theorem

- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge
 - ❖ The update stops after making a finite number of mistakes.
 - ❖ The convergence rate depends on **the difficulty of the problem**

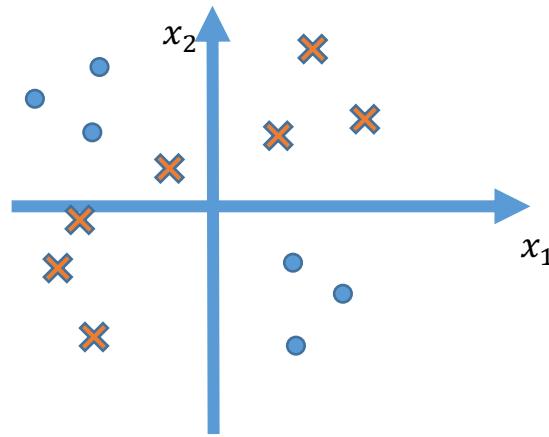


Convergence theorem

- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is **linearly separable**), the perceptron algorithm will converge.
- ❖ Note: this is the condition of the data we may not know what the hyperplane is



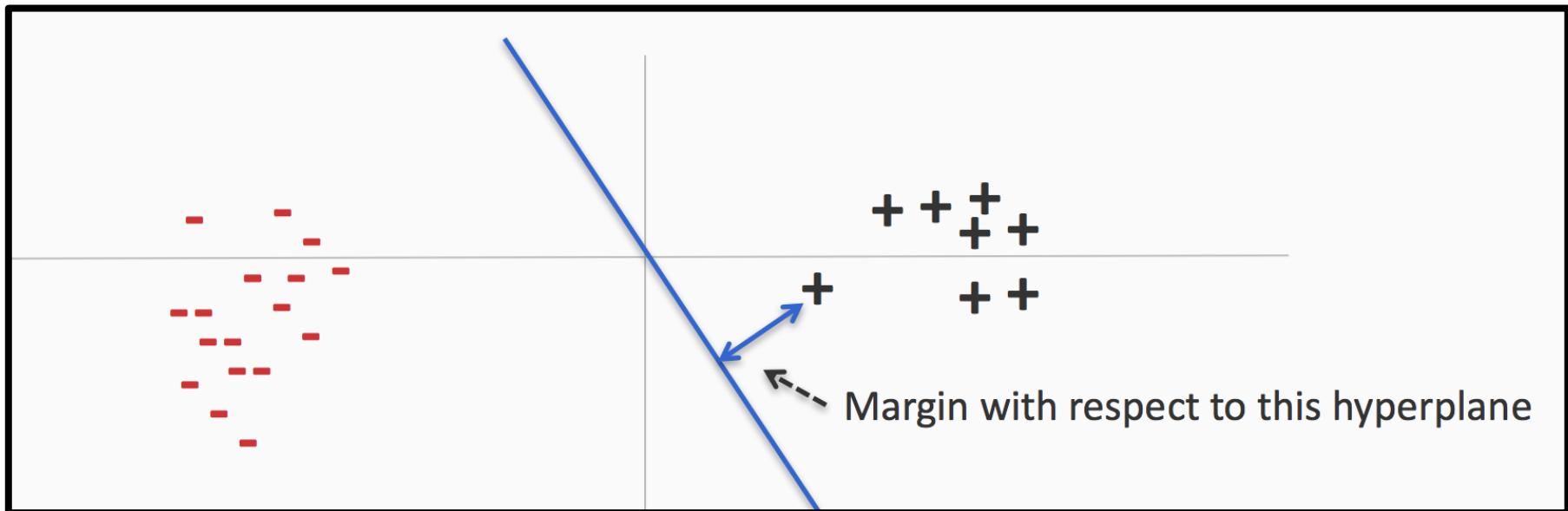
Linearly separable data



Linearly non-separable data

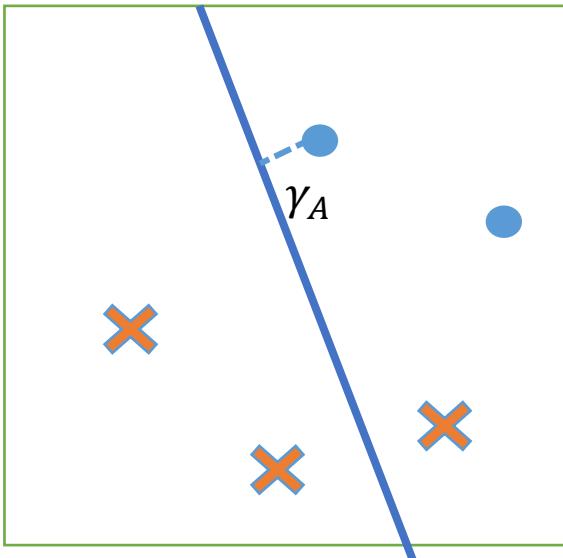
Margin

If a hyperplane can separate the data. The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

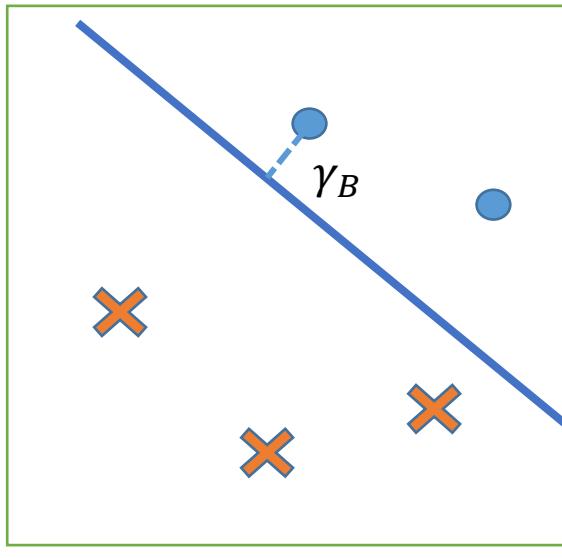


Margin

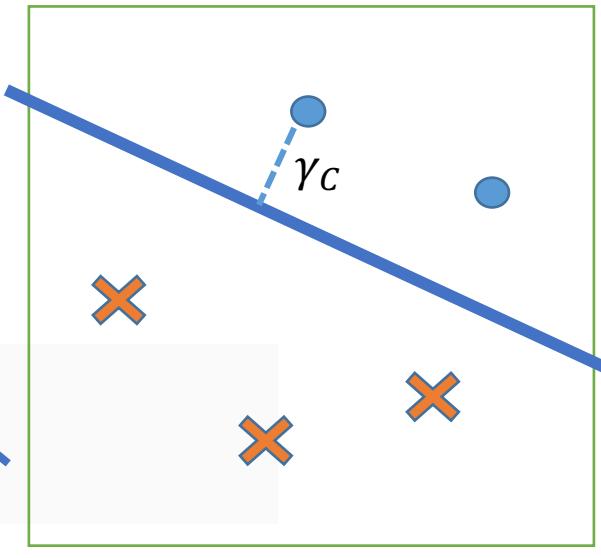
- ❖ If a hyperplane can separate the data. The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- ❖ The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.
Which γ is the margin of data?



(a)



Lec 6: Perceptron
(b)

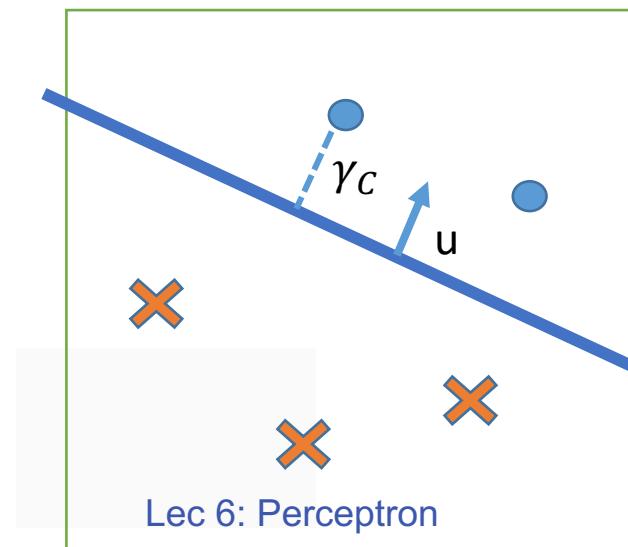


(c)

56

Margin

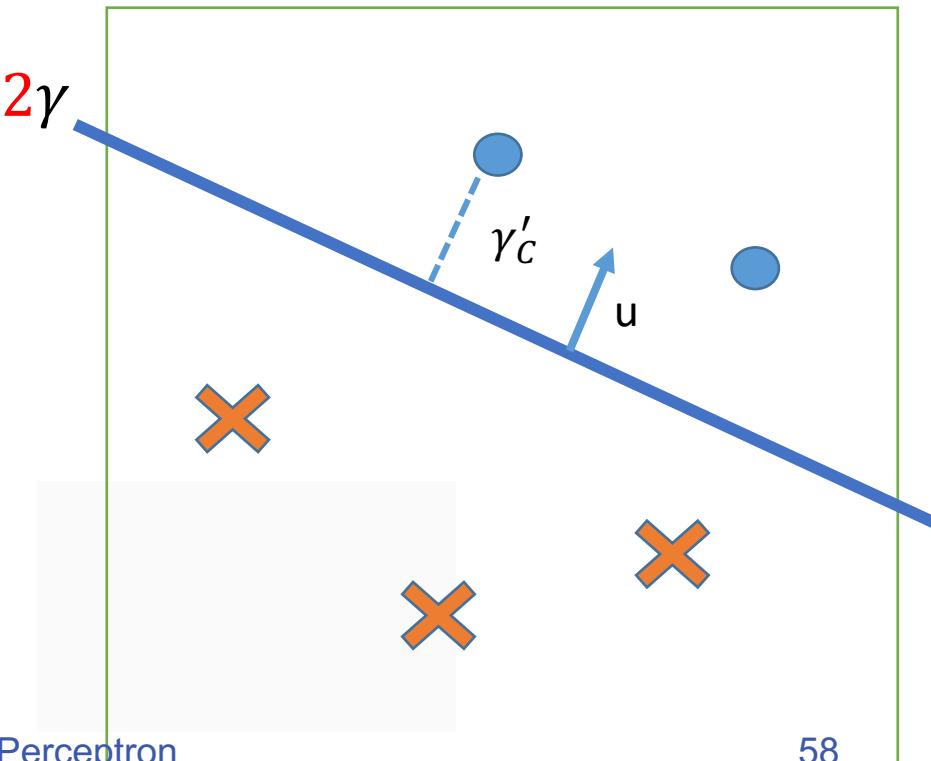
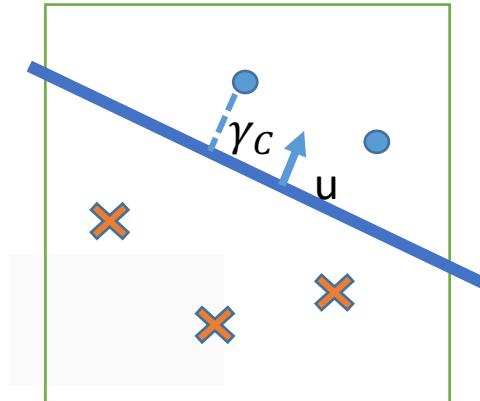
- ❖ The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.
- ❖ Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ be a set of training data, if there exists a unit vector \mathbf{u} such that $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$, for all data points (x_i, y_i) in the training set
- ❖ γ is the margin



Margin is not scale invariance

- ❖ Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ be a set of training data, if there exists a unit vector \mathbf{u} such that $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$, for all data points (\mathbf{x}_i, y_i) in the training set
- ❖ If we double the size of every input \mathbf{x}_i , the margin γ is also double, because

$$y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma \Rightarrow y_i (\mathbf{u}^\top 2\mathbf{x}_i) \geq 2\gamma$$



Margin is not scale invariance

- ❖ The “difficulty” of the problem, can be captured by $\frac{R}{\gamma}$,
 $\|\mathbf{x}_i\| \leq R, \forall i$
- ❖ Perceptron makes $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes if data has margin γ and
the size of all the input vectors $\|\mathbf{x}_i\| \leq R, \forall i$

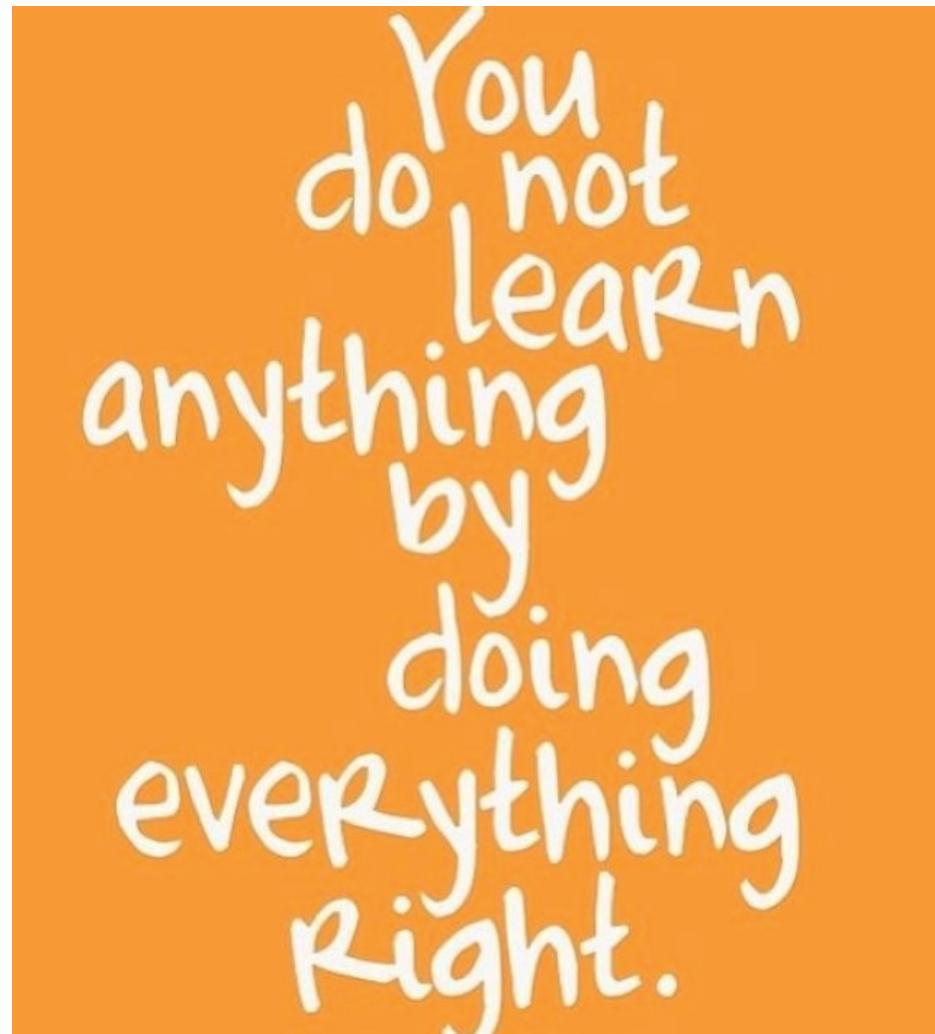
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ be a sequence of training examples such that for all i , the feature vector $x_i \in R^n$, $\|x_i\| \leq R$ and the label $y_i \in \{-1, +1\}$.

Suppose there exists a unit vector $u \in R^n$ (i.e $\|u\| = 1$) such that for some $\gamma > 0$ we have $y_i (u^\top x_i) \geq \gamma$.

Then, the perceptron algorithm will make at most $(R/\gamma)^2$ mistakes on the training sequence.

NOTE!! # mistakes!= # seen data points



Beyond the separable case

- ❖ Good news
 - ❖ Perceptron makes no assumption about data distribution, could be **even adversarial**
- ❖ Bad news:
 - ❖ Real-world data are often **not** linearly separable

What you learned today

- ❖ Linear models
- ❖ The Perceptron Algorithm
- ❖ Perceptron Mistake Bound

Not cover in the lecture and the exams

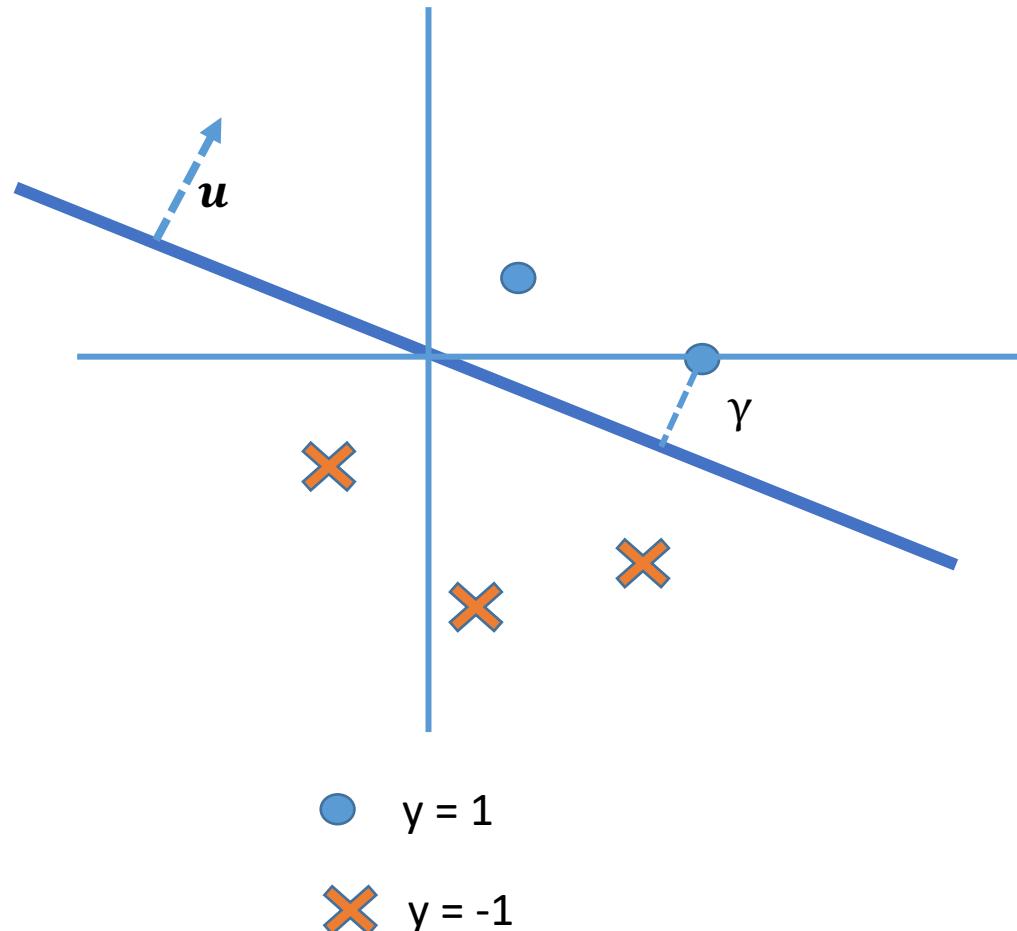
Appendix: Proof of mistake bound

See details at

<https://arxiv.org/pdf/1305.0208.pdf>

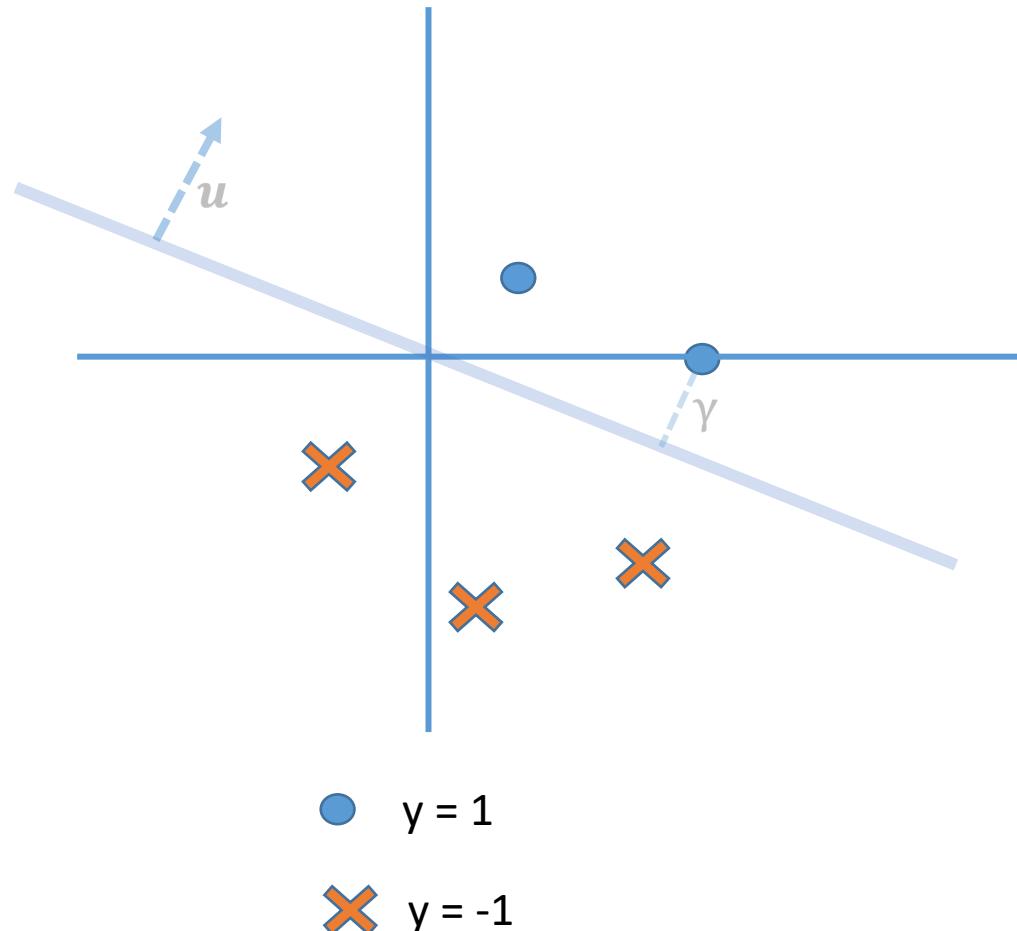
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



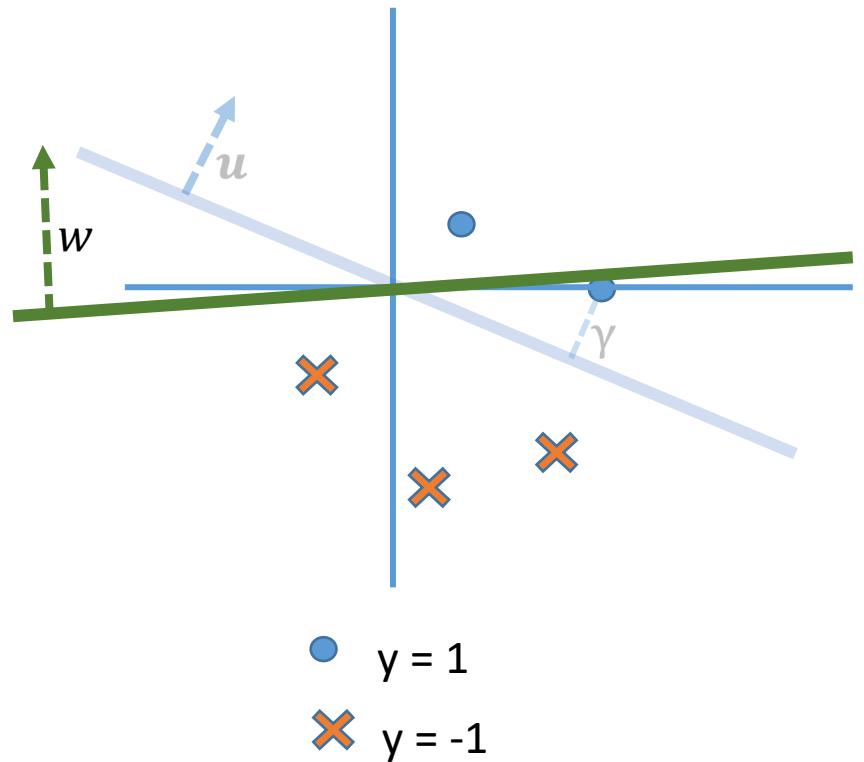
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



Intuition

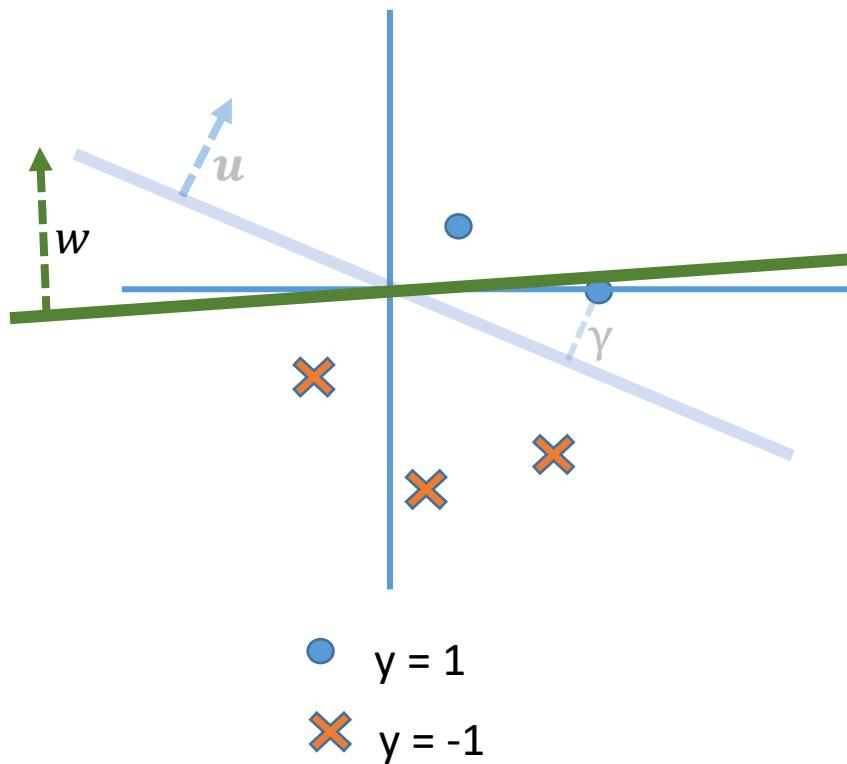
$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



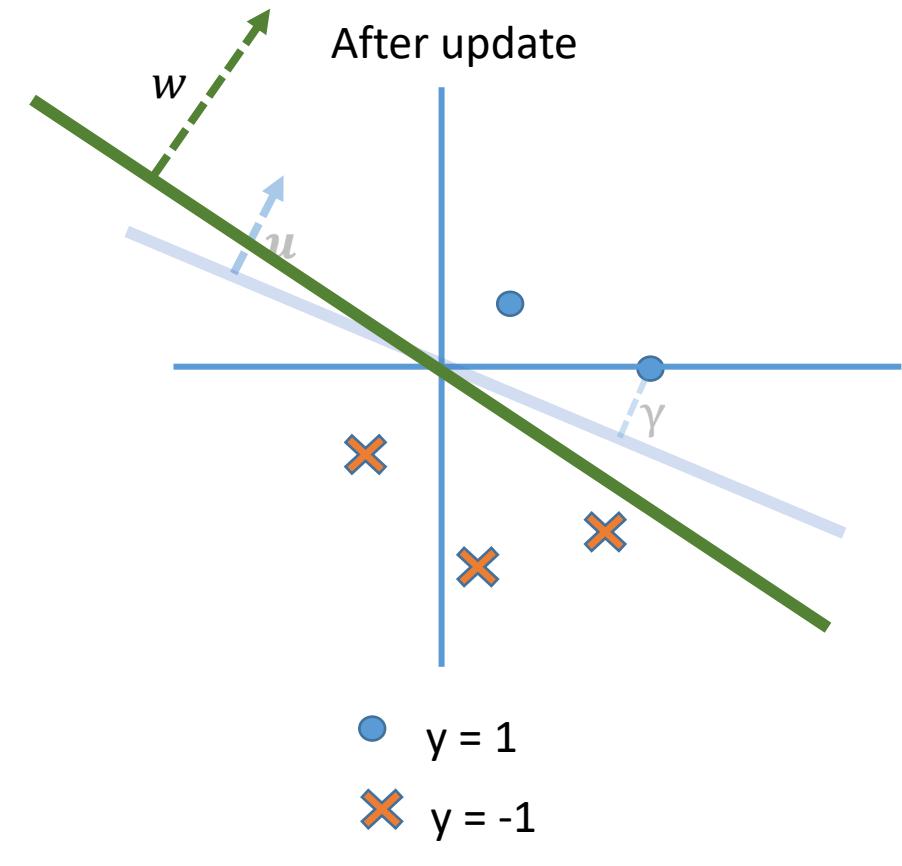
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$

Before update



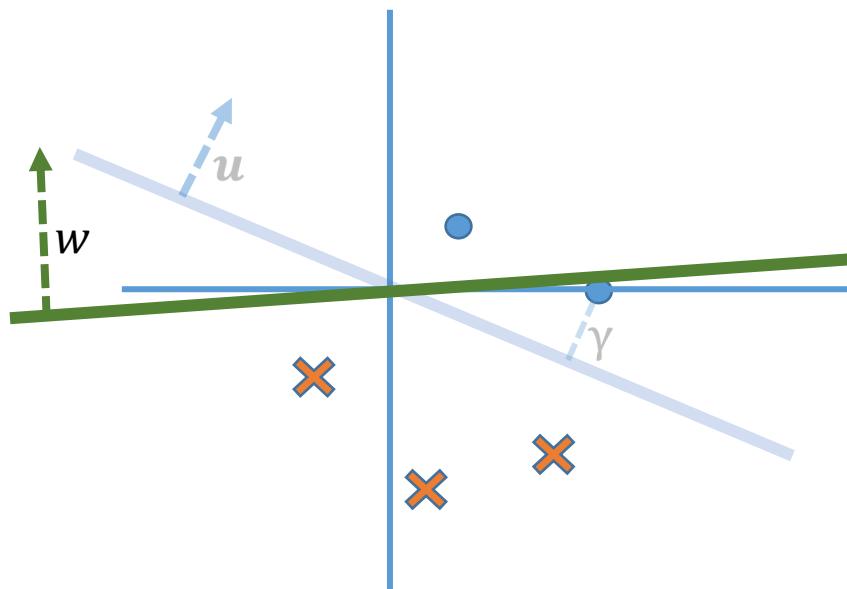
After update



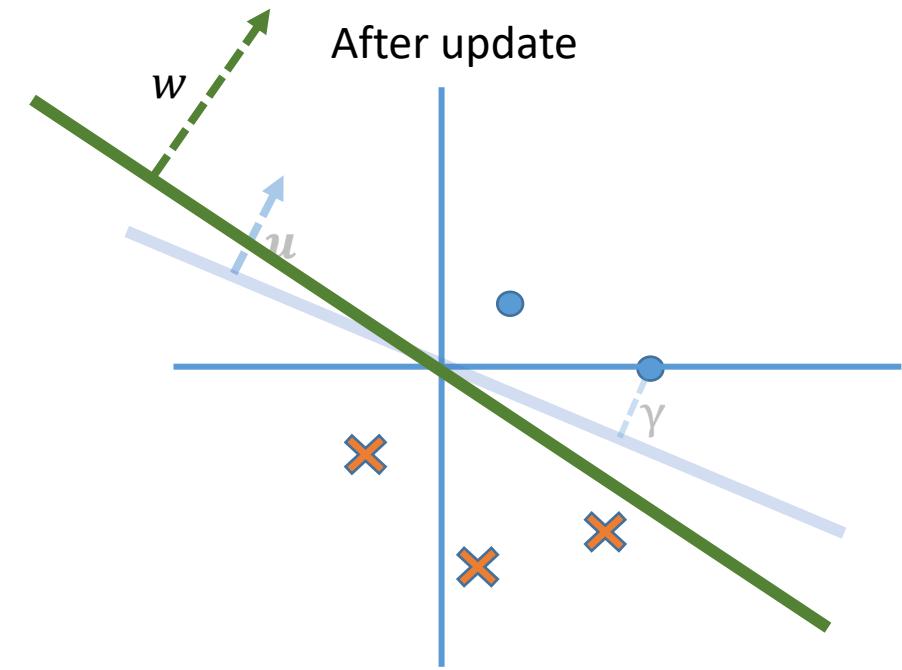
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$

Before update



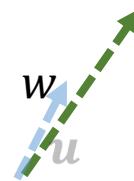
After update



Before update

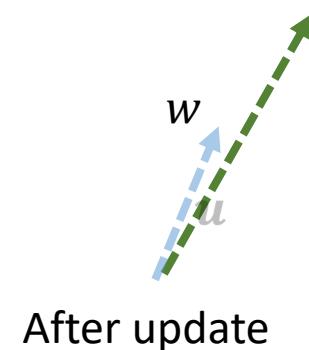
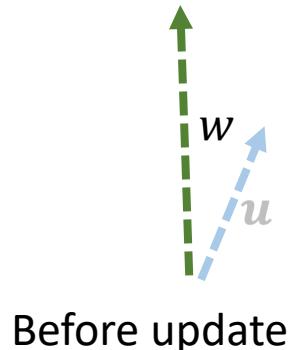
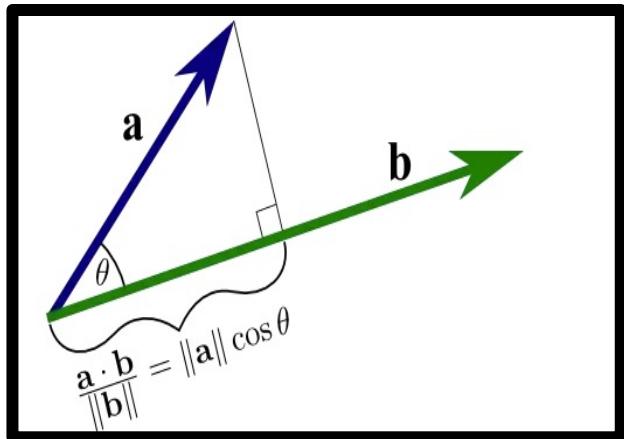


After update



Intuition

1. After update, $\mathbf{u}^T \mathbf{w}_{t+1}$ is larger than $\mathbf{u}^T \mathbf{w}_t$
After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t \gamma$
2. The size of $\|\mathbf{w}_{t+1}\|$ may increase, but not much
After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$



Proof (preliminaries)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

The setting

- ❖ Initial weight vector \mathbf{w} is all zeros
- ❖ All training examples are contained in a ball of size R
 - ❖ $\|\mathbf{x}_i\| \leq R$
- ❖ The training data is separable by margin γ using a unit vector \mathbf{u}
 - ❖ $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$

Proof (1/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

1. Claim: After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t \gamma$

$$\begin{aligned}\mathbf{u}^T \mathbf{w}_{t+1} &= \mathbf{u}^T \mathbf{w}_t + y_i \mathbf{u}^T \mathbf{x}_i \\ &\geq \mathbf{u}^T \mathbf{w}_t + \gamma\end{aligned}$$

Because the data is
separable by a
margin γ
 $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^\top \mathbf{w}_0 = 0$), straightforward induction gives us

$$\mathbf{u}^\top \mathbf{w}_t \geq t \gamma$$

Intuition: the inner product between the underlying true model and the current model is non-decreasing after each update
1). The directions of u, w align or 2). $\|\mathbf{w}\|$ increases

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^T \mathbf{x}_i) + \|\mathbf{x}_i\|^2\end{aligned}$$

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^\top \mathbf{x}_i) + \|\mathbf{x}_i\|^2\end{aligned}$$

The weight is updated only when there is a mistake.
That is when $y_i \mathbf{w}_t^\top \mathbf{x}_i < 0$.

$\|\mathbf{x}_i\| \cdot R$, by definition of R

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^\top \mathbf{x}_i) + \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + R^2\end{aligned}$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^\top \mathbf{w}_0 = 0$),
straightforward induction gives us $\|\mathbf{w}_t\|^2 \leq tR^2$

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

Intuition: $\|w\|$ is bounded

$$\begin{aligned}\|\mathbf{w}_{t+1}\| &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^\top \mathbf{x}_i) + \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + R^2\end{aligned}$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^\top \mathbf{w}_0 = 0$),
straightforward induction gives us $\|\mathbf{w}_t\|^2 \leq tR^2$

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

Intuition 1: the inner product between the underlying true model and the current model is non-decreasing after each update
1). The directions of u, w align or 2). $\|w\|$ increases

Intuition 2: $\|w\|$ is bounded

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\|$$

From (2)

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t$$

From (2)



$$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\| \|\mathbf{w}_t\| \cos(\text{angle between them})$$

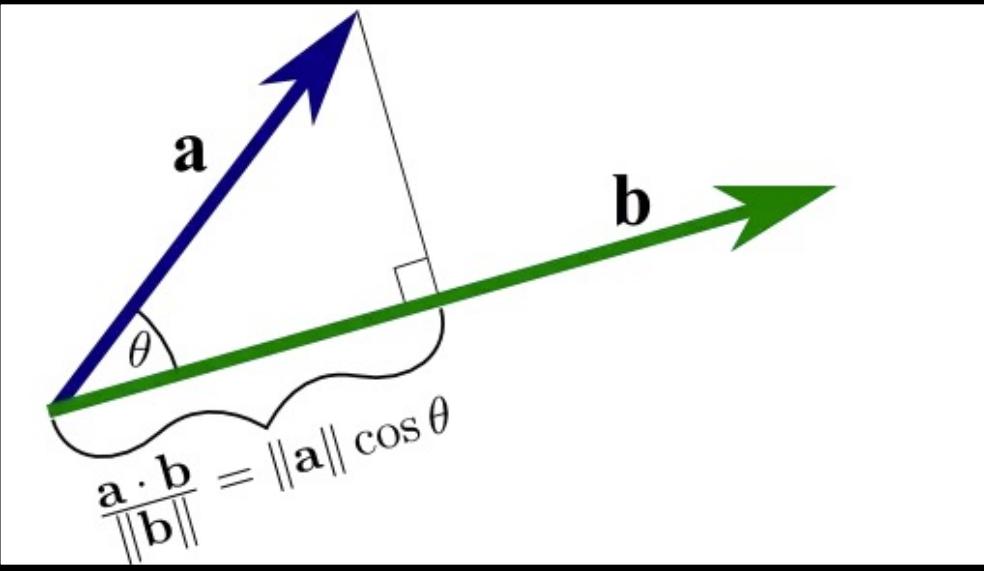
But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ *(Cauchy-Schwarz inequality)*

Proof (

What we

1. After
2. After



$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t$$

From (2)



$$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\| \|\mathbf{w}_t\| \cos(\text{angle between them})$$

But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ (Cauchy-Schwarz inequality)

Lec 6: Perceptron

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^\top \mathbf{w}_t \geq t\gamma$$

From (2)

From (1)

Proof (3/3)

mistakes != # seen data points

What we know:

1. After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$



$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t \geq t\gamma$$

Number of mistakes $t \leq \frac{R^2}{\gamma^2}$

Bounds the total number of mistakes!

Lecture 7: Logistic Regression

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu, Hal Daume whose slides are heavily used, and the many others who made their course material freely available online.

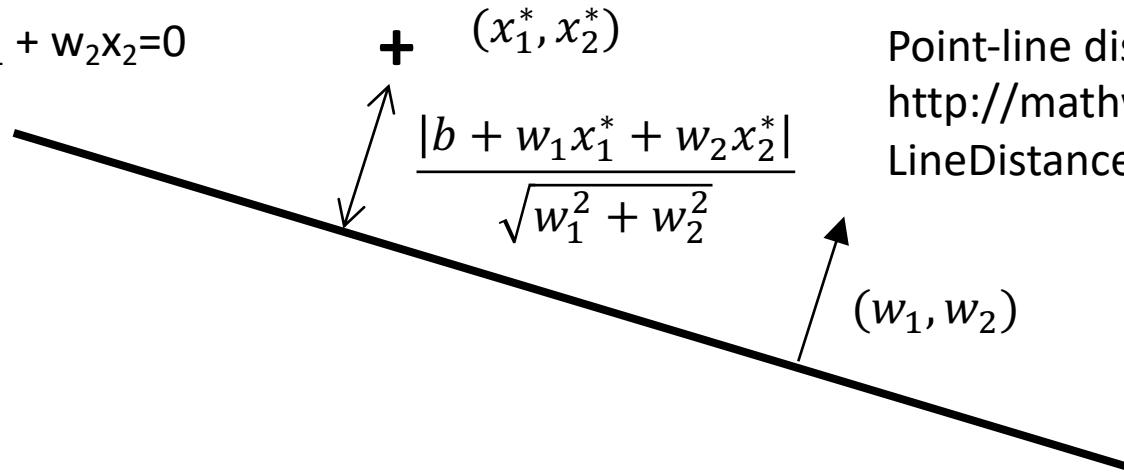
Announcement

- ❖ Quiz 2 is coming!
 - ❖ Suggested readings in BruinLearn
<http://ciml.info/>
 - ❖ Please don't put CM146 course materials online
-
- ❖ Explanation of margin γ
$$y_i(\mathbf{u}^T \mathbf{x}_i + b) \geq \gamma$$

Recall: The geometry of a linear classifier

$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$

$$b + w_1 x_1 + w_2 x_2 = 0$$

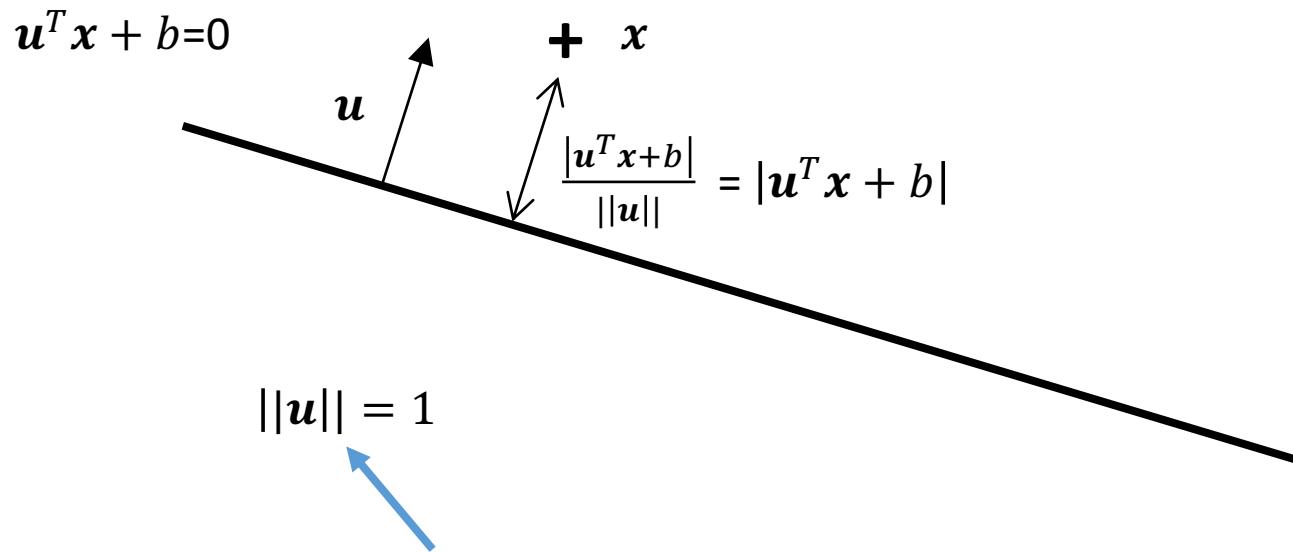


Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Recall: The geometry of a linear classifier

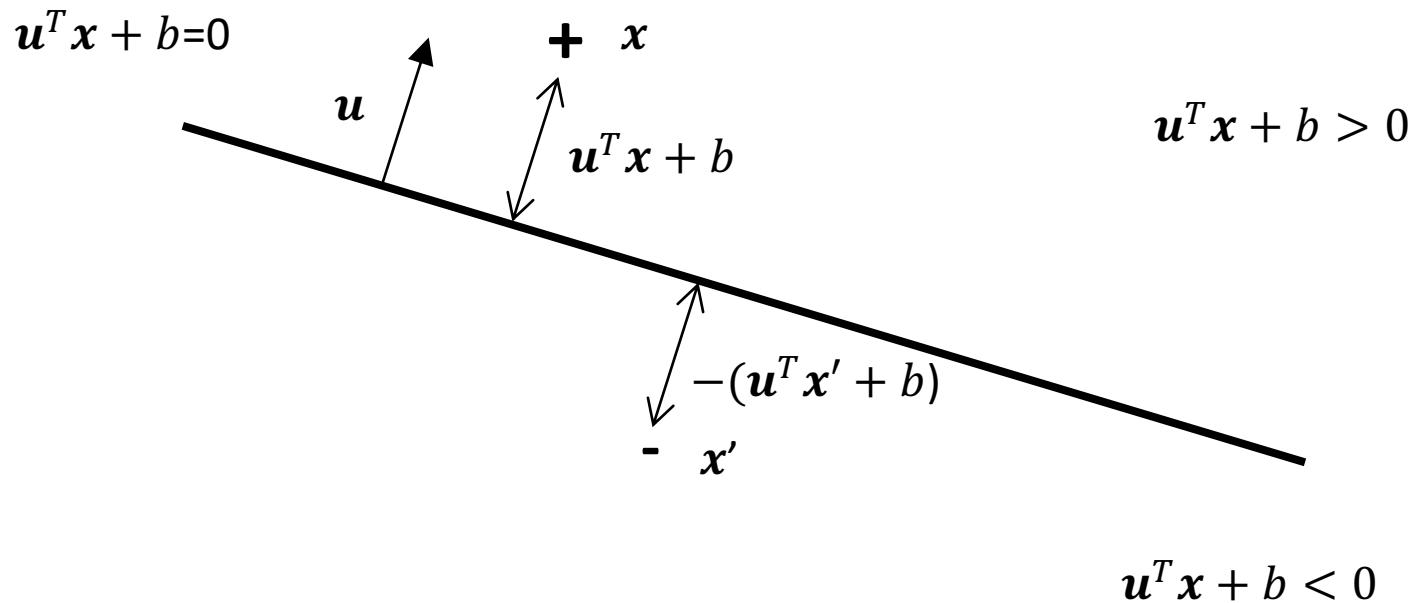
$$\text{Prediction} = \text{sgn}(\mathbf{u}^T \mathbf{x} + b)$$



If we use a **unit** normal vector \mathbf{u} represents the hyperplane, the distance between point \mathbf{x} to plane is $|\mathbf{u}^T \mathbf{x} + b|$ or $y(\mathbf{u}^T \mathbf{x} + b)$

Recall: The geometry of a linear classifier

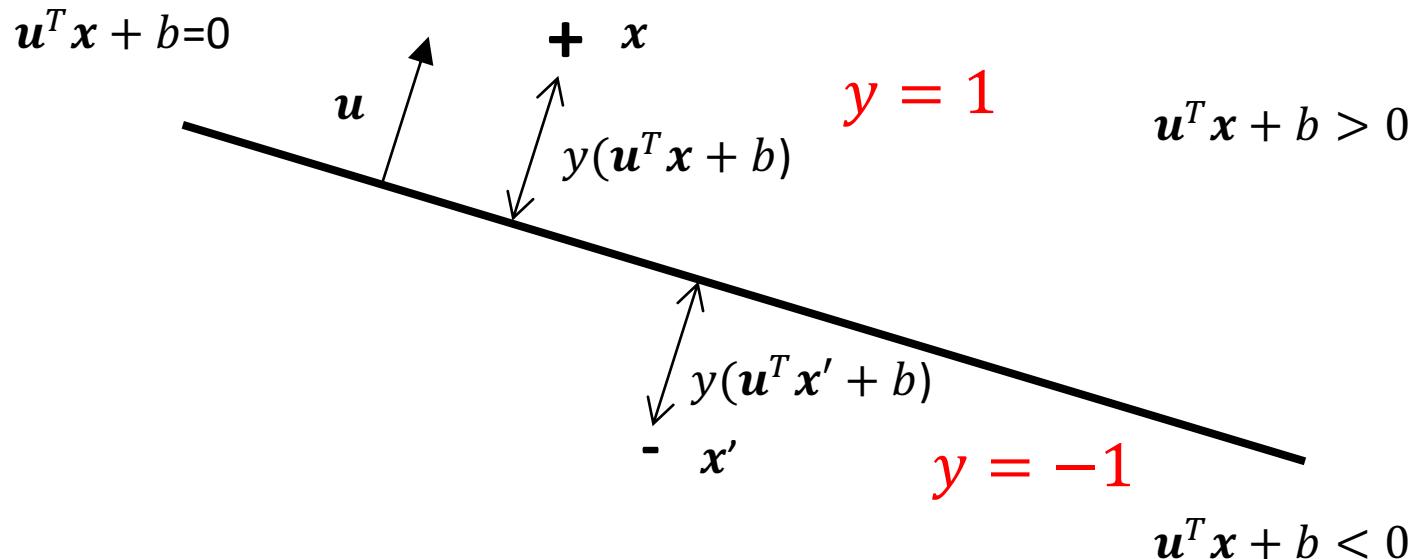
$$\text{Prediction} = \text{sgn}(\mathbf{u}^T \mathbf{x} + b)$$



If we use a **unit** normal vector \mathbf{u} represents the hyperplane, the distance between point \mathbf{x} to plane is $|\mathbf{u}^T \mathbf{x} + b|$ or $y(\mathbf{u}^T \mathbf{x} + b)$

Recall: The geometry of a linear classifier

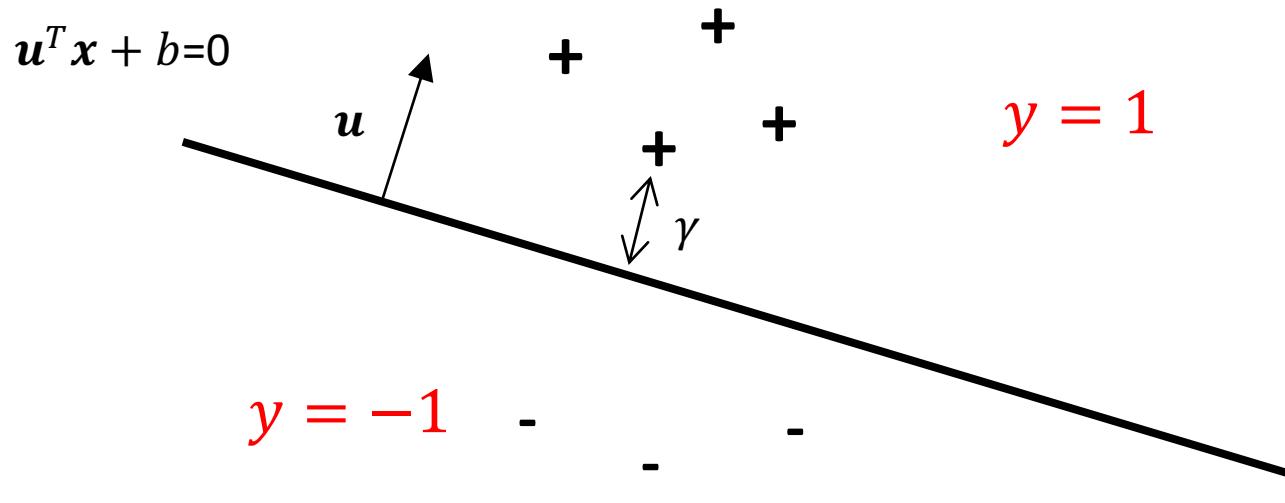
$$\text{Prediction} = \text{sgn}(\mathbf{u}^T \mathbf{x} + b)$$



If we use a **unit** normal vector \mathbf{u} represents the hyperplane, the distance between point \mathbf{x} to plane is $|\mathbf{u}^T \mathbf{x} + b|$ or $y(\mathbf{u}^T \mathbf{x} + b)$

Recall: The geometry of a linear classifier

$$\text{Prediction} = \text{sgn}(\mathbf{u}^T \mathbf{x} + b)$$



If we use a **unit** normal vector \mathbf{u} represents the hyperplane, the distance between point \mathbf{x} to plane is $|\mathbf{u}^T \mathbf{x} + b|$ or $y(\mathbf{u}^T \mathbf{x} + b)$

If the distance between the closest point in dataset D to the plane \mathbf{u} is γ

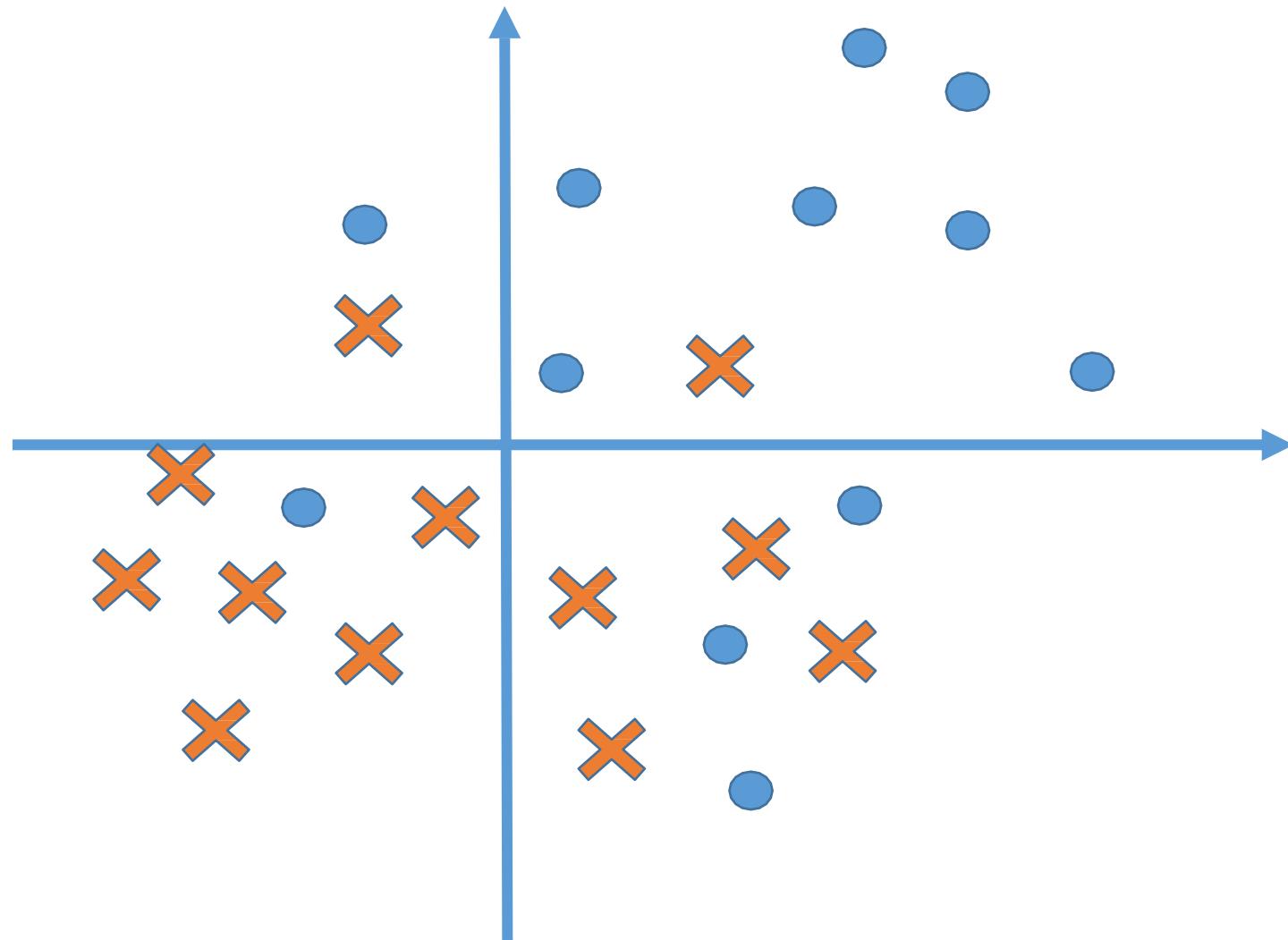
$$y_i(\mathbf{u}^T \mathbf{x}_i + b) \geq \gamma, \forall (\mathbf{x}_i, y_i) \in D$$

Logistic regression

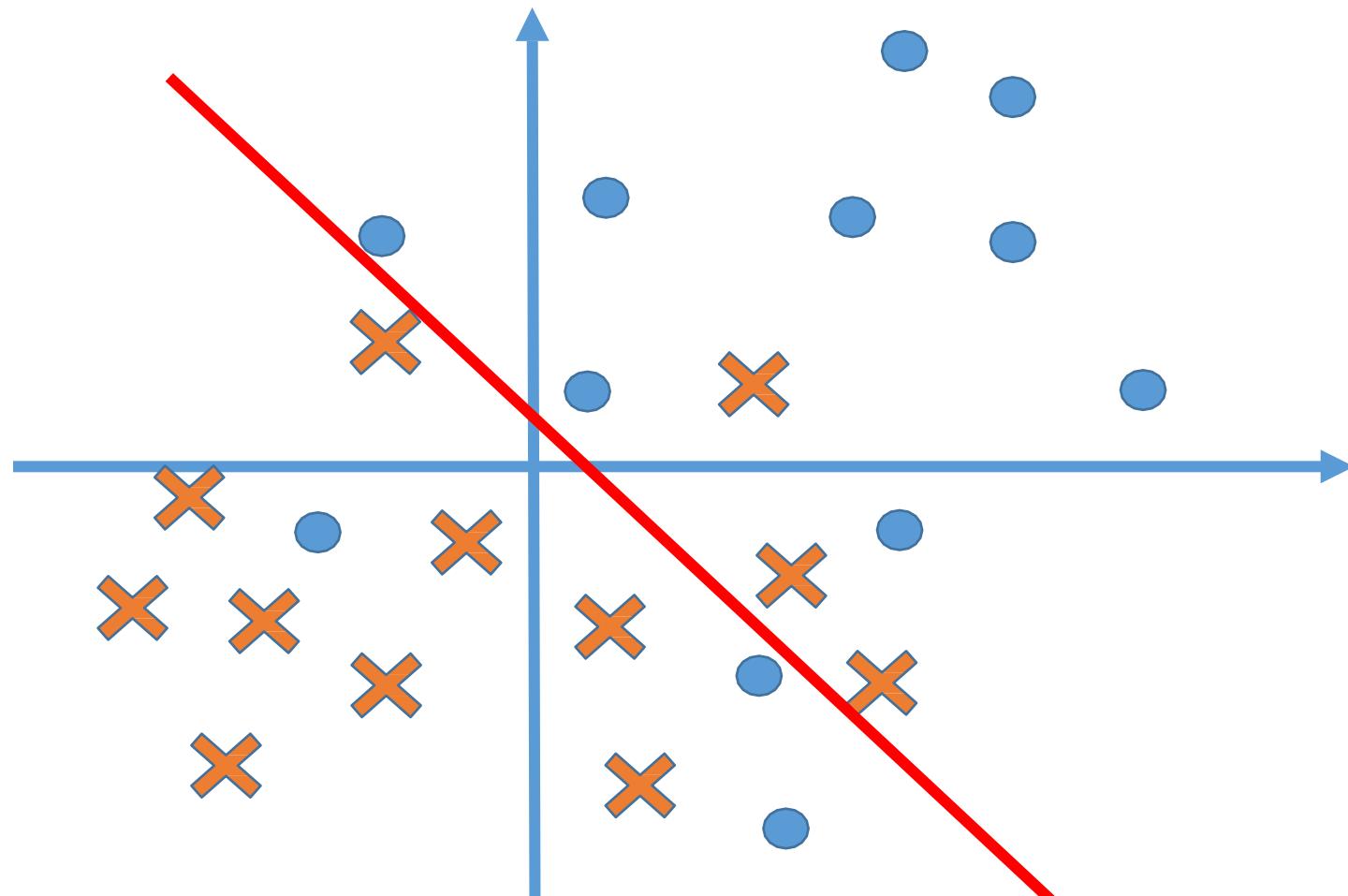
What you will learn today

- ❖ Logistic regression assumption
- ❖ Sigmoid function
- ❖ Maximum likelihood principle
- ❖ Optimization in ML
 - ❖ Stochastic gradient decent

What if data is not linearly separable?

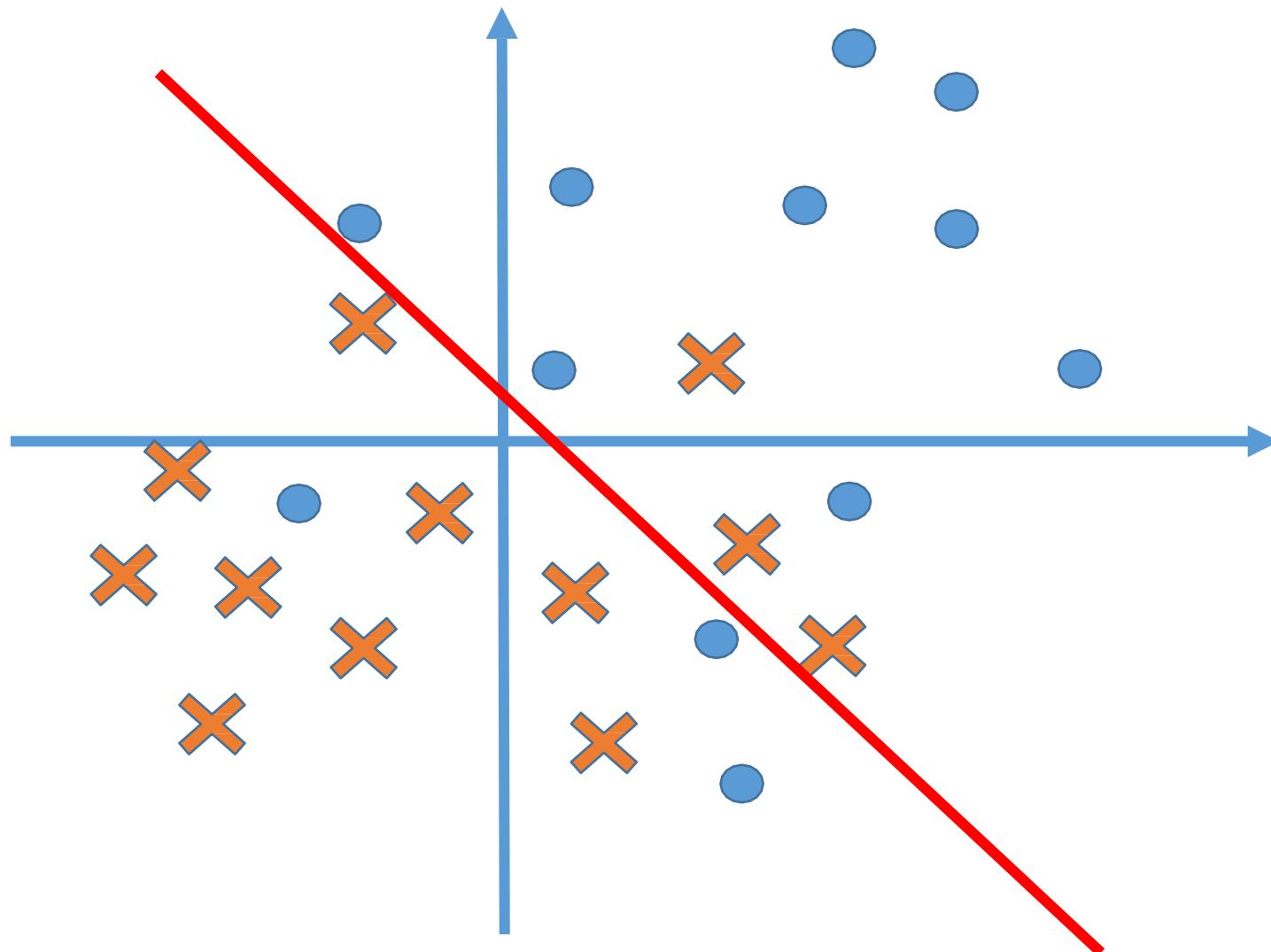


What if data is not linearly separable?



There is no linear model can separate all the data well

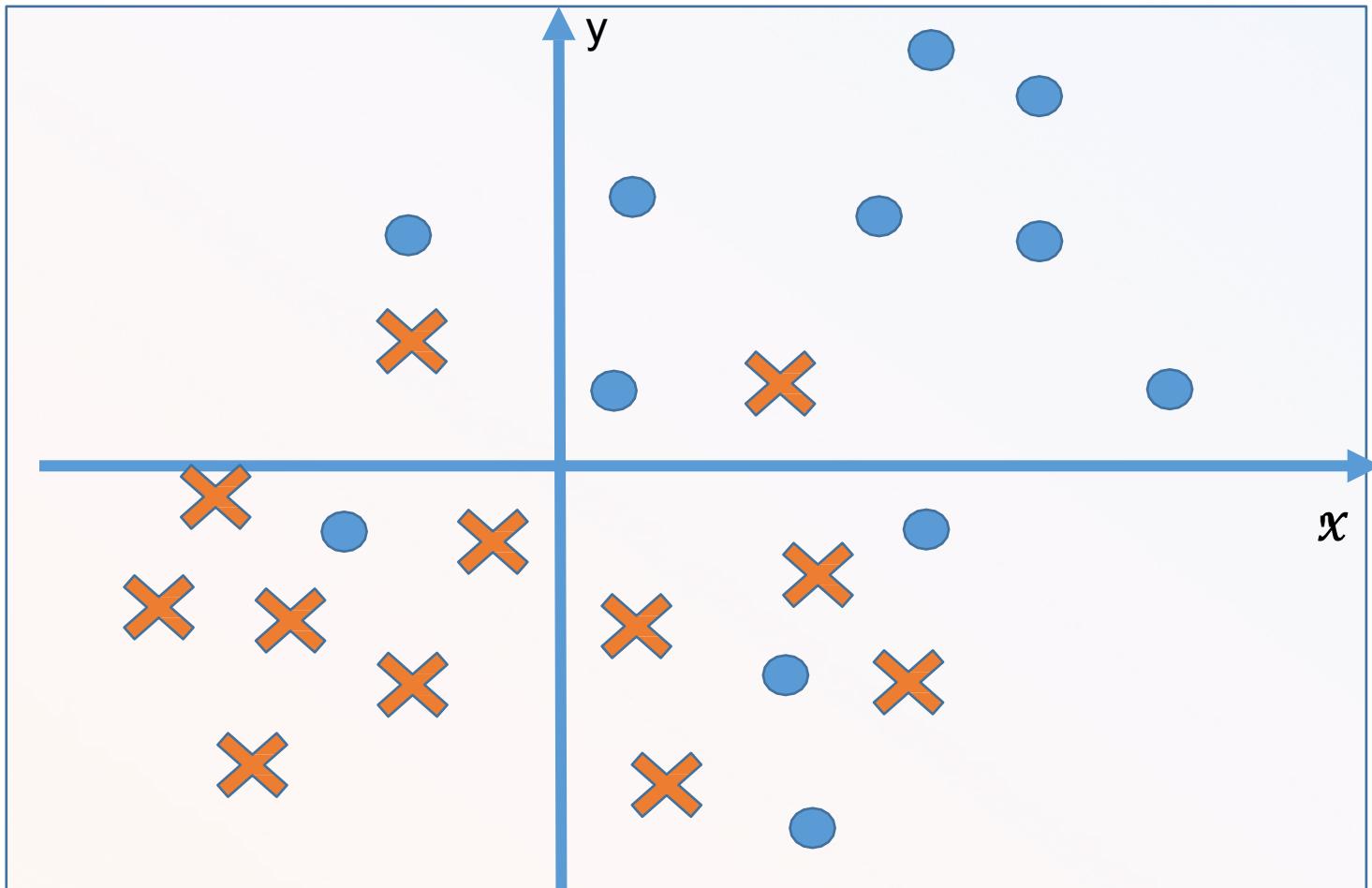
What Making data not linearly separable? [Discussion]



What Making data not linearly separable?

- ❖ Decision boundary is nonlinear
 - ❖ E.g., XOR example
- ❖ Noise in the training data
 - ❖ Outlier due to annotation errors
- ❖ Not enough features
- ❖ The nature of the prediction task
 - ❖ Patients with the same lab test results may not always have the same disease

What if data is not linearly separable?

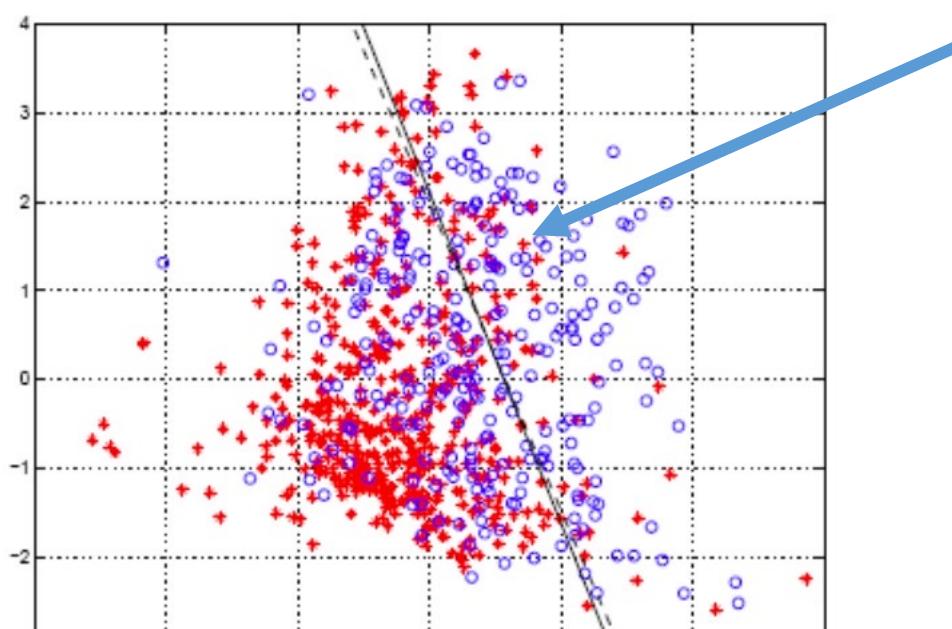


We consider a probabilistic model (today's lecture)

Modeling the Probability

Classification, but...

- ❖ The output y is a discrete value
- ❖ Instead of predicting the output label,
let's predict $P(y = 1 | x)$

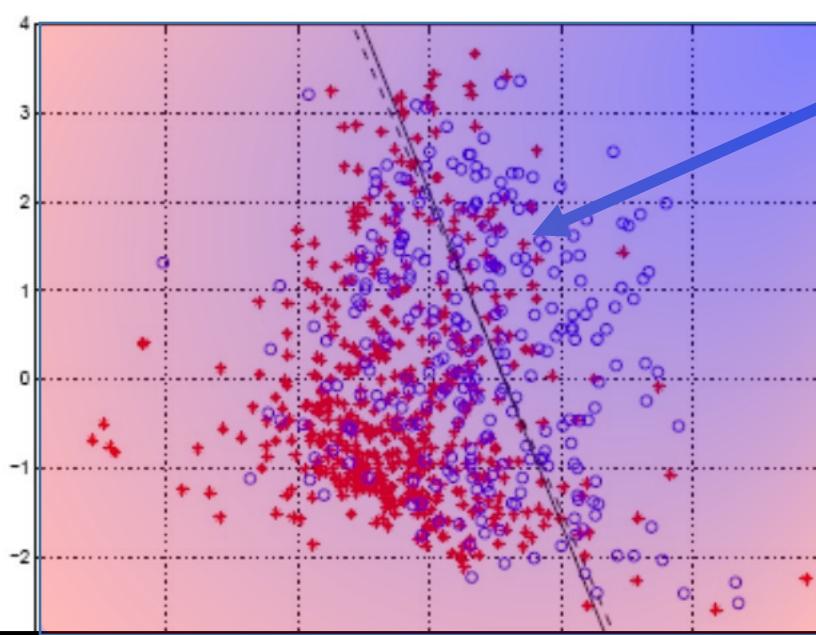


How likely the label $y = 1$
if my feature vector is
in this region

Perceptron does not produce probability estimates

Classification, but...

- ❖ The output y is a discrete value
- ❖ Instead of predicting the output label, let us predict $P(y = 1 | \mathbf{x})$



How likely the label $y = 1$
if my feature vector is
in this region

Perceptron does not produce probability estimates

Predict $P(y = 1 \mid \mathbf{x})$

Input: $x \in \mathbb{R}^d$

Output: $y \in \{1, -1\}$

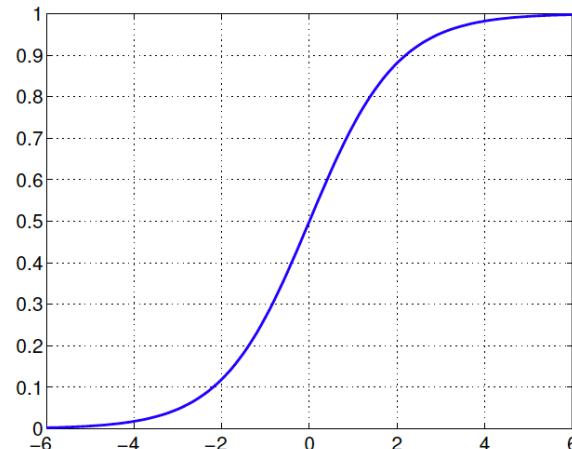
Build a model $h(x)$ such that

$$h(x) = \sigma(w^T x + b) \approx P(y = 1|x)$$

a regression problem

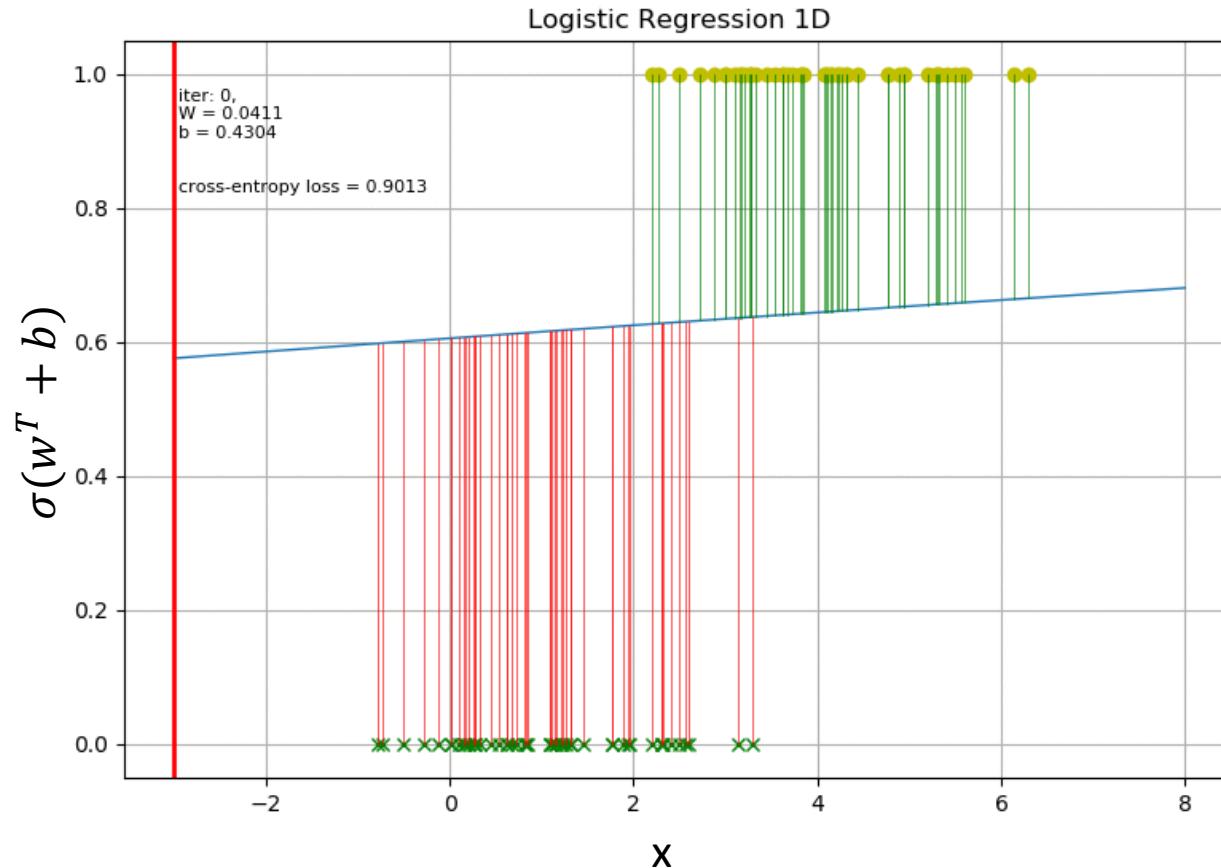
σ is a sigmoid function

$$\begin{aligned}\sigma(z) &= \frac{\exp(z)}{1 + \exp(z)} \\ &= \frac{1}{1+\exp(-z)}\end{aligned}$$



Logistic Regression in Action

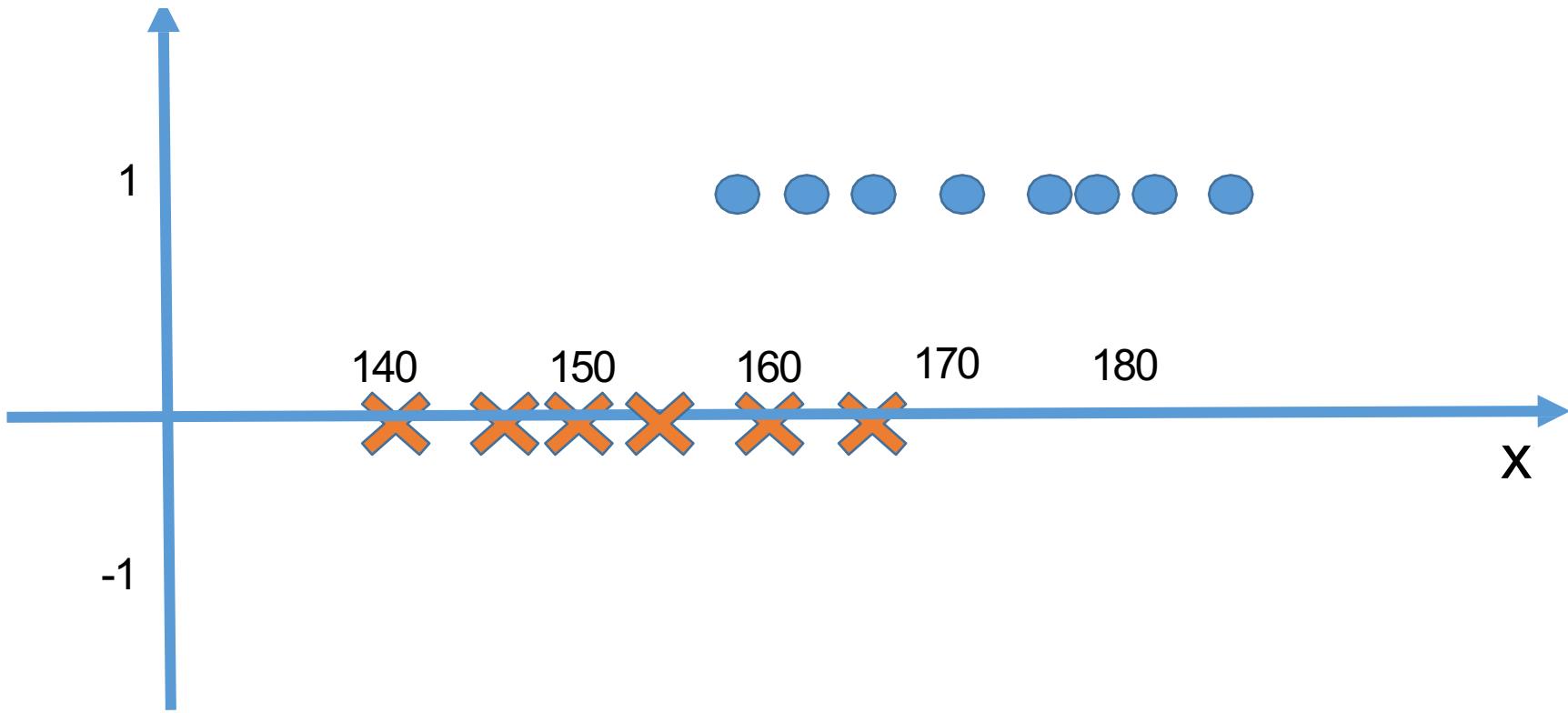
$$\sigma(w^T x + b) \approx P(y = 1|x)$$



<https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf>

Why sigmoid function?

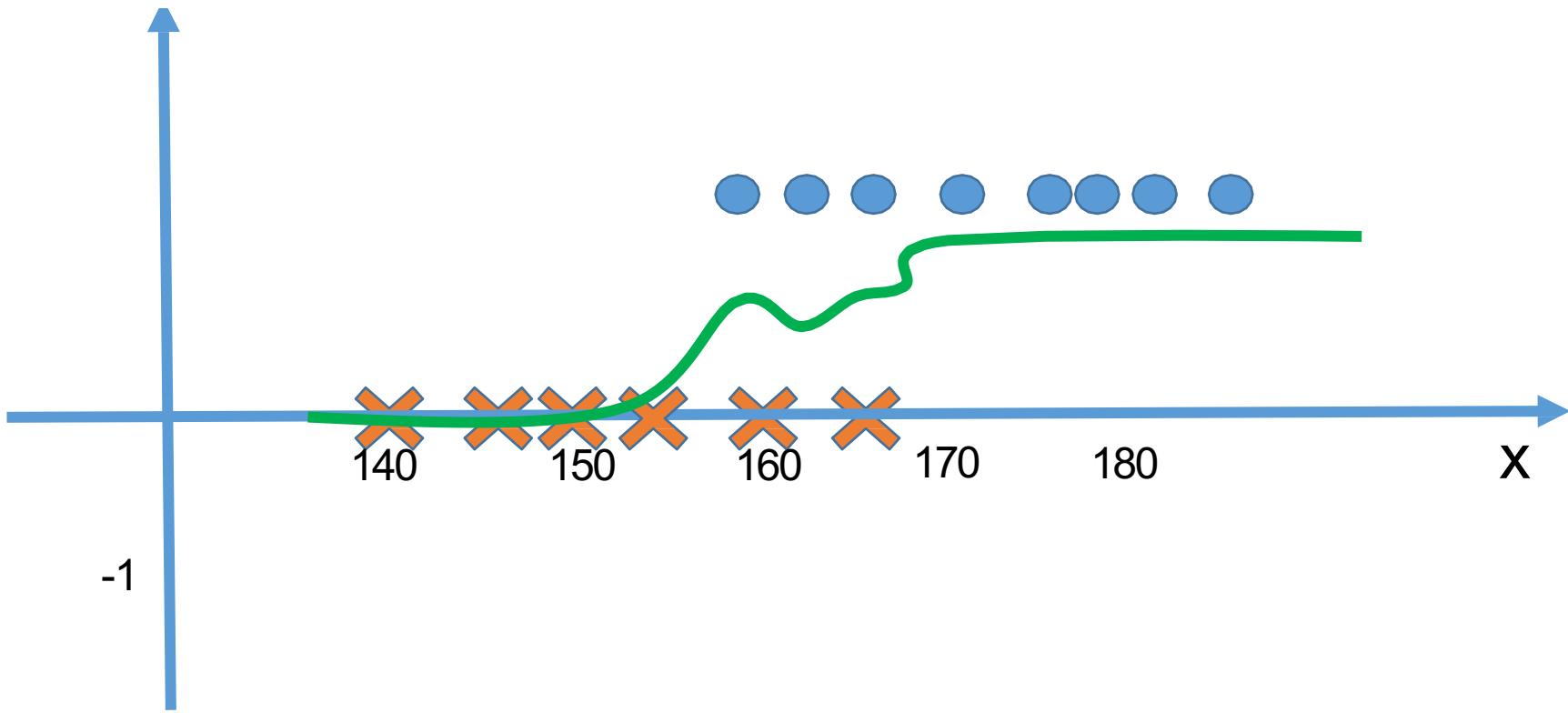
$$P(y = 1|x)$$



What is the σ function

What is the underlying target function?

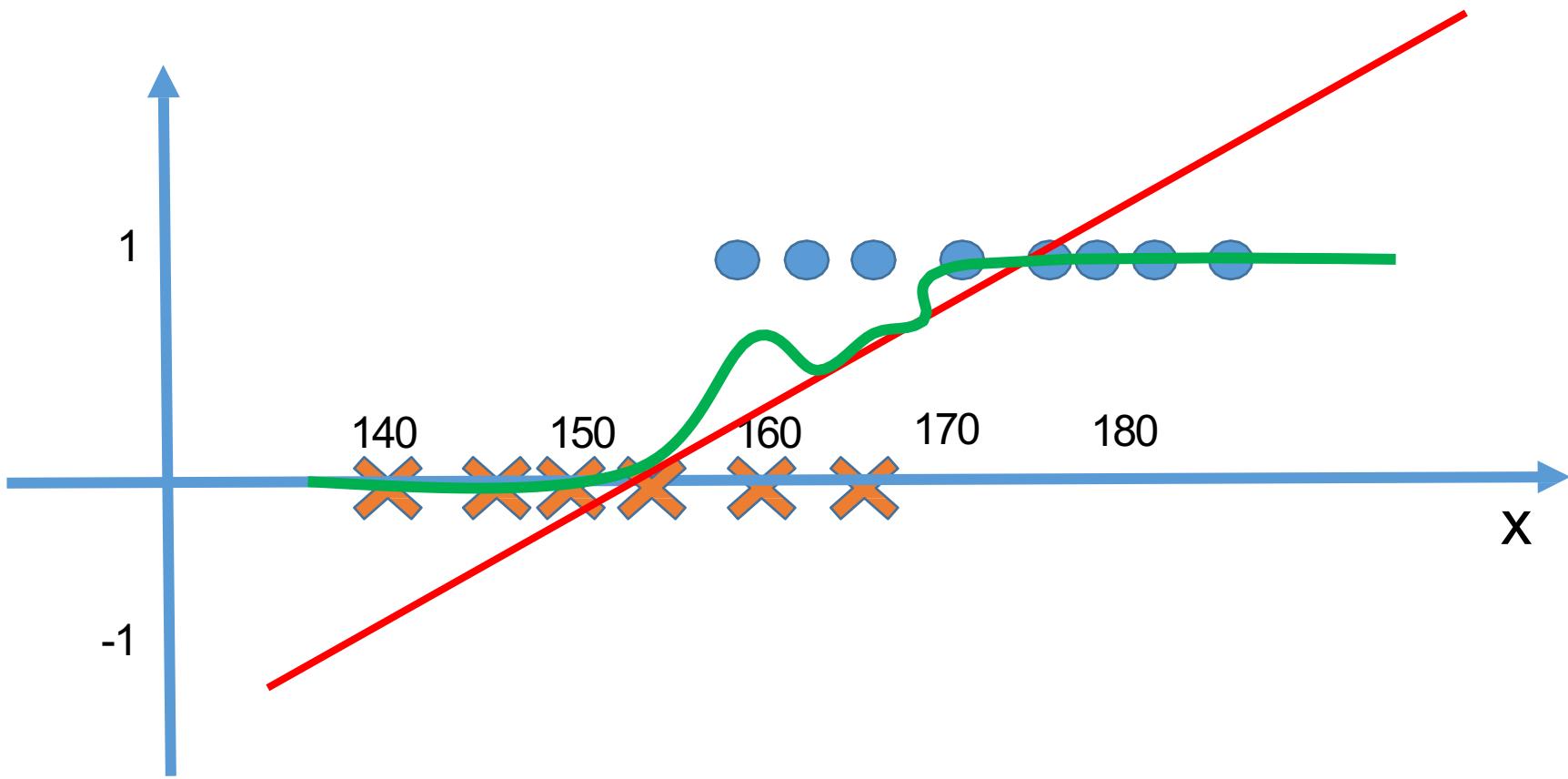
$$P(y = 1|x)$$



How to fit $P(y = 1|x)$

Can we fit it with a linear function?

$$y = \mathbf{w}^T \mathbf{x} + b$$



How to fit $P(y = 1|x)$

Probability cannot be larger than 1

1

140

150

160

170

180

-1

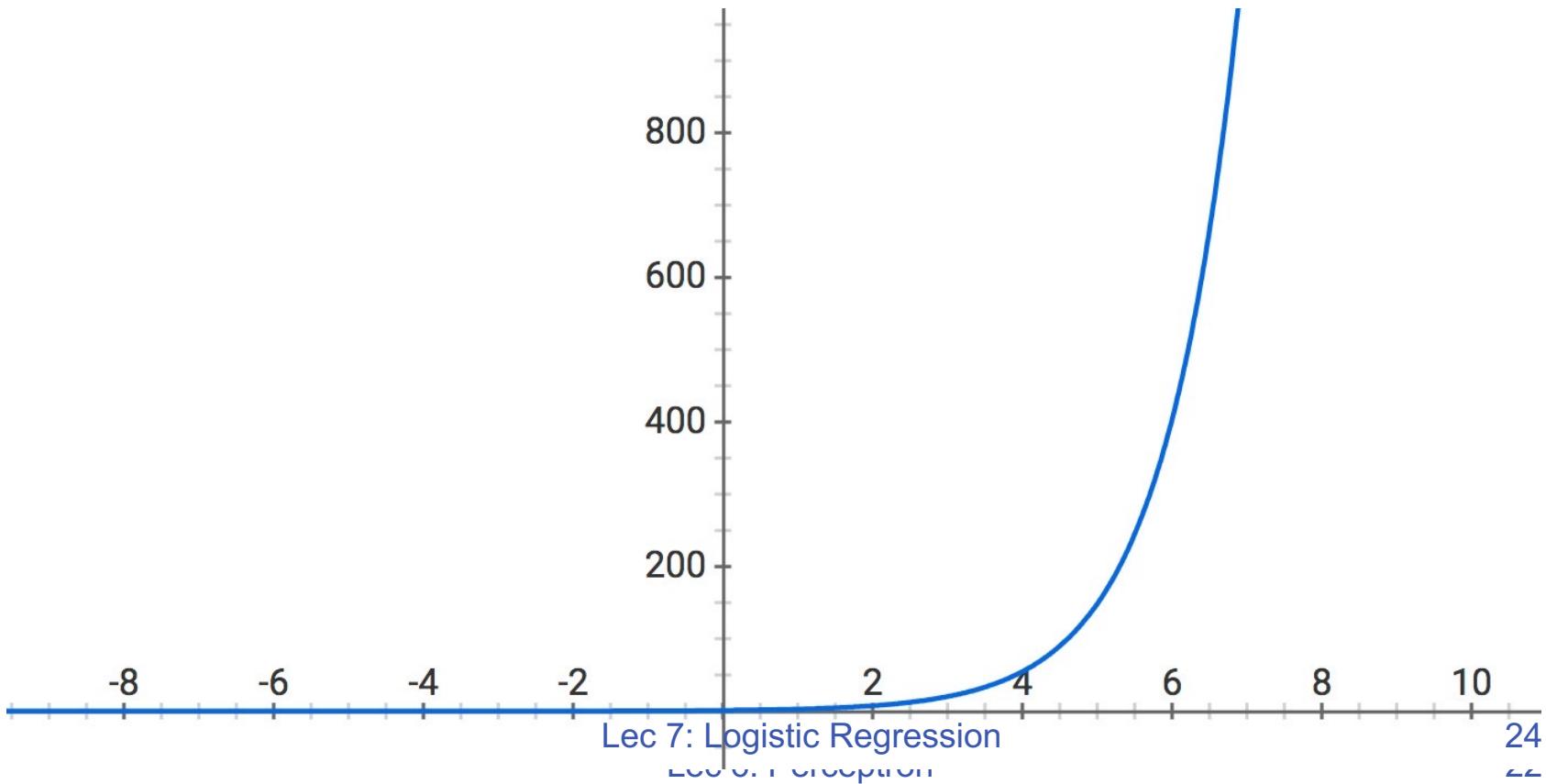
x

Probability cannot be negative

How can we design such a transformation function?

Idea 1: function always output positive value

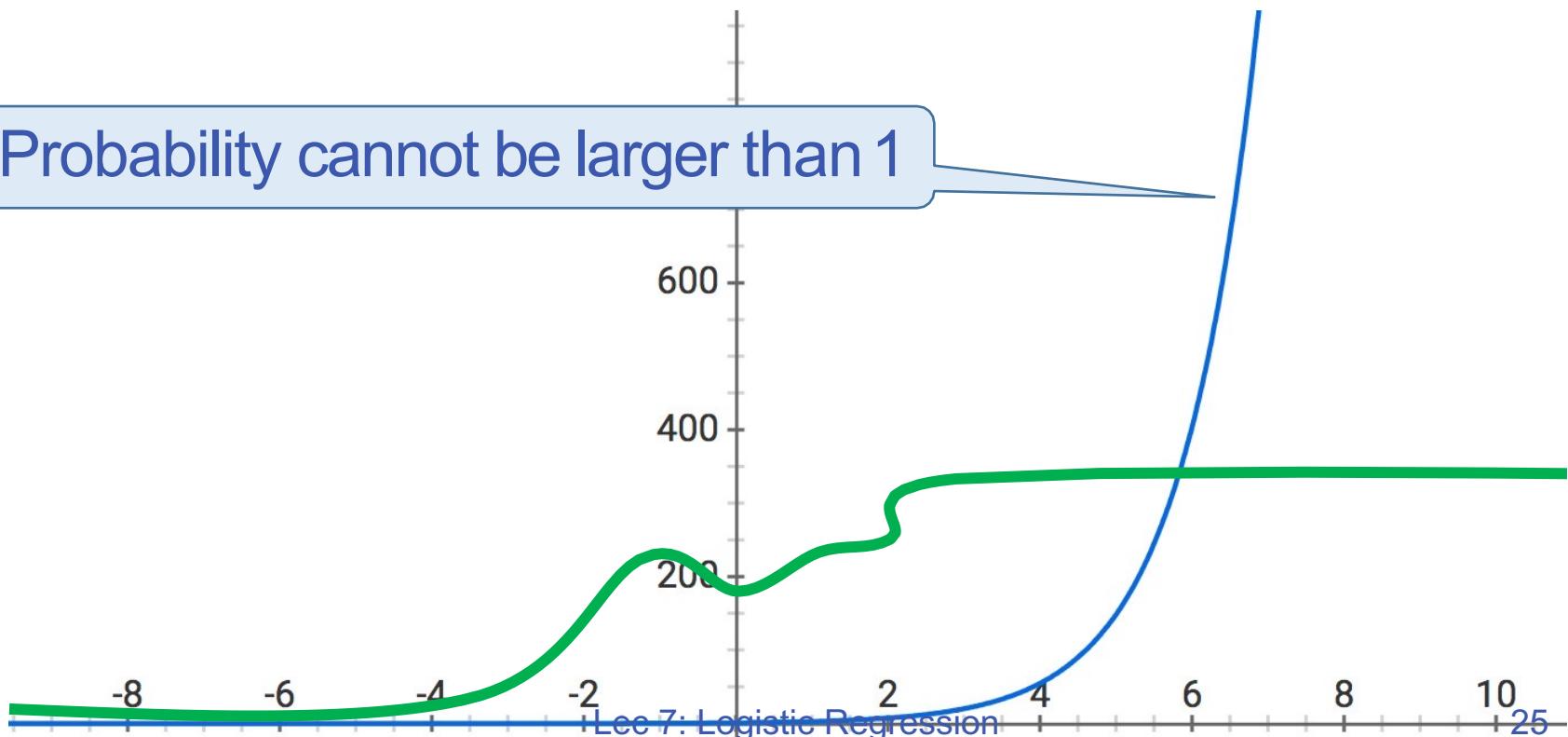
$\exp(\cdot)$ is always positive



How can we design such a transformation function?

- ❖ Idea 1: function always output positive value
 - ❖ $\exp(\cdot)$ is always positive
 - ❖ $\exp(w^T x + b)$ always return positive value

Probability cannot be larger than 1



How can we design such a transformation function?

Idea 2: normalize the value such that it is less than 1

- ❖ $\exp(w^T x + b) \in (0, \infty)$ grows very fast, so we need to use exp to normalize itself
- ❖ Let's use

$$\sigma(w^T x + b) = \frac{\exp(w^T x + b)}{1 + \exp(w^T x + b)}$$

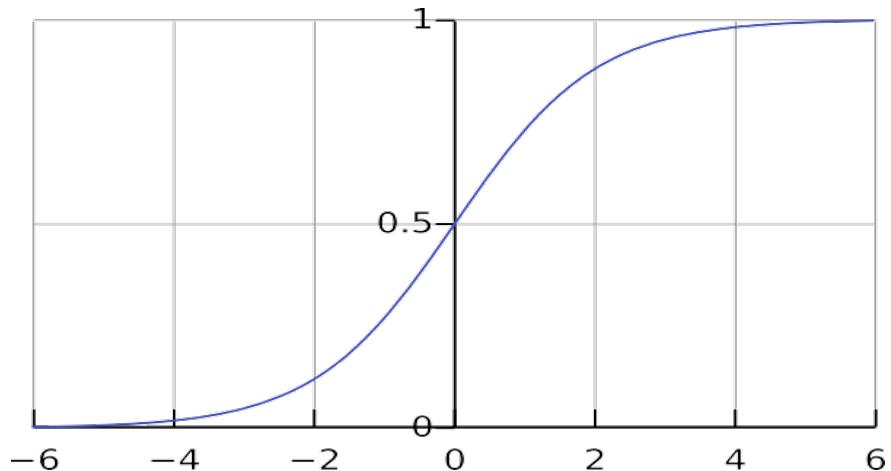
- ❖ When $w^T x + b \rightarrow \infty$,
 $\sigma(w^T x + b) \rightarrow 1$
- ❖ When $w^T x + b \rightarrow -\infty$,
 $\sigma(w^T x + b) \rightarrow 0$



The Sigmoid function

The $\sigma(z)$ function is called sigmoid function
(or logistic function)

$$\begin{aligned}\sigma(z) &= \frac{\exp(z)}{1 + \exp(z)} \\ &= \frac{1}{1+\exp(-z)}\end{aligned}$$



Summary (Modeling)

- ❖ What is the goal of logistic regression?
 - ❖ Model $P(y = 1|x)$
- ❖ What is the hypothesis space?

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

We want to find $h(x)$ such that

$$h(x) \approx P(y = 1 \mid x)$$

Decision Boundary of Logistic Regression

Predicting a label

$$P(y = 1 | \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

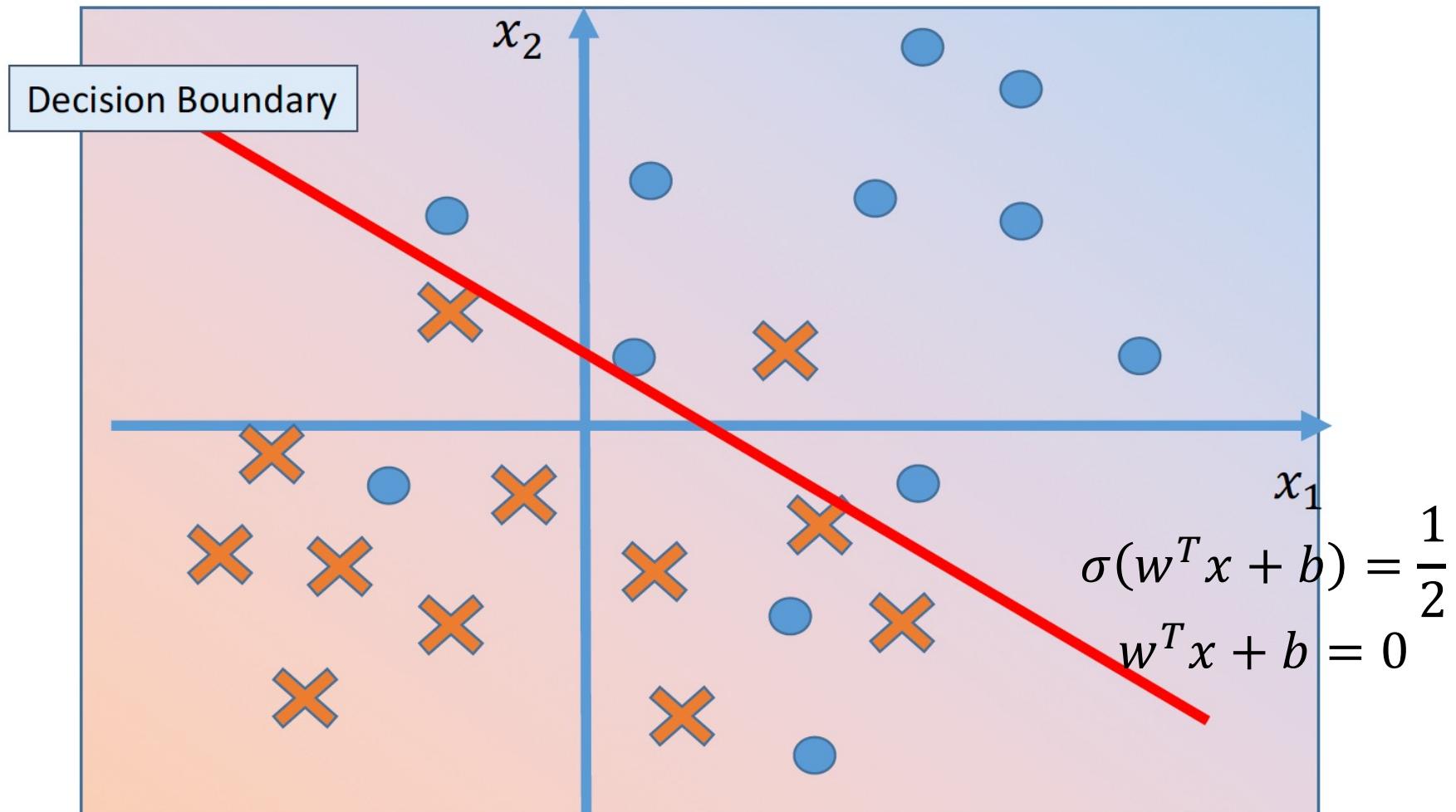
Compute $\sigma(w^T x)$;

If this is greater than half, predict 1
else predict -1

What does this correspond to in terms of $\mathbf{w}^T \mathbf{x}$?

$$w^T x = 0$$

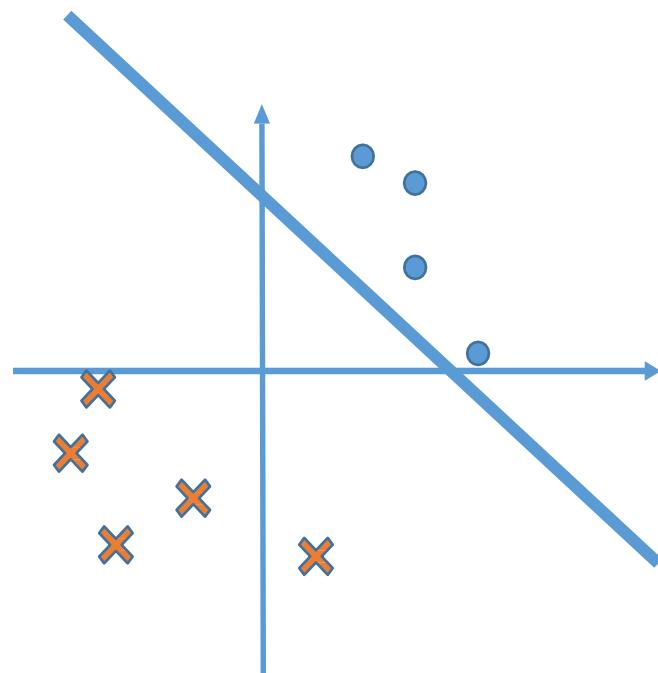
Prediction by logistic regression



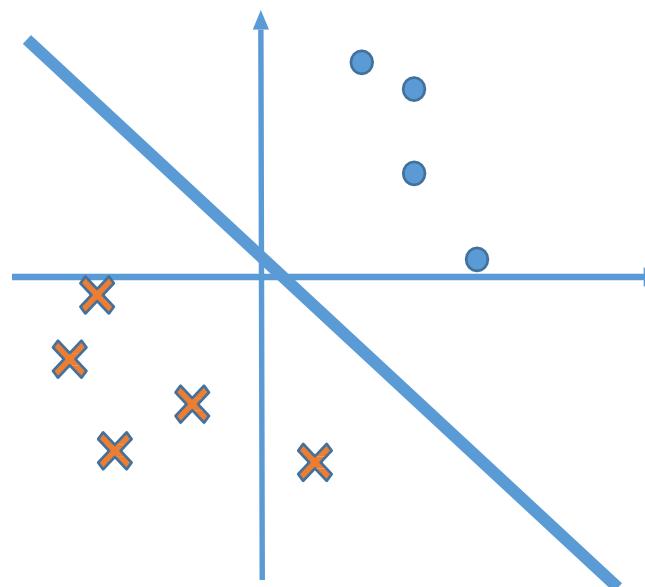
A probabilistic model of modeling $\sigma(w^T x + b) = P(y = 1|x)$

Exercise

- ❖ Which function(s) are likely to be a decision function of logistic regression



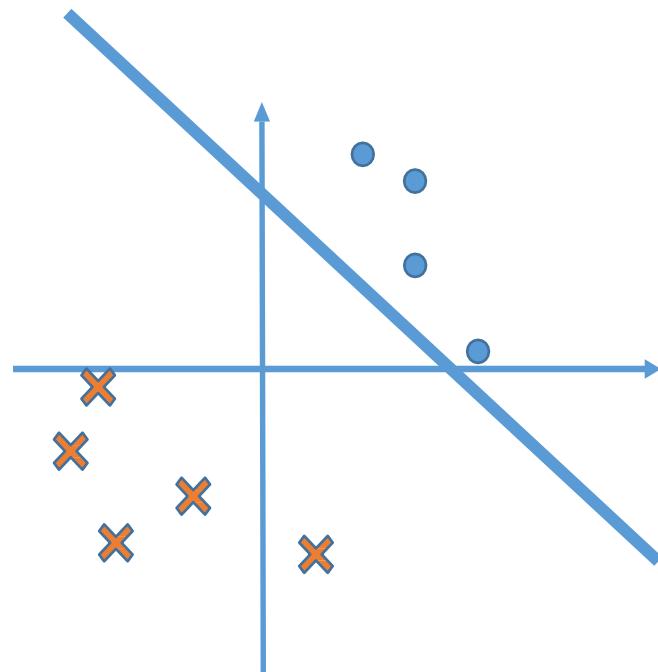
(A)



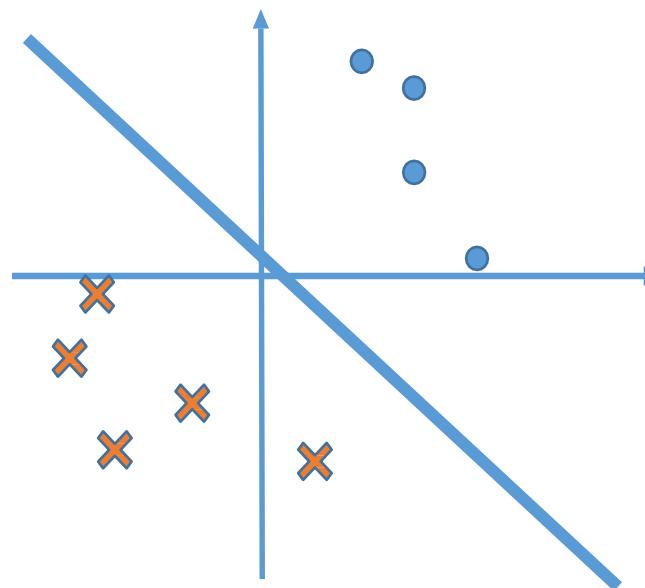
(B)

Exercise

- ❖ Which function(s) are likely to be a decision function of Perceptron



(A)



(B)

How to Train a Logistic Regression Model?

Logistic Regression: Setup

- ❖ The setting
 - ❖ Binary classification
 - ❖ Inputs: Feature vectors $x \in R^N$
 - ❖ Labels: $y \in \{-1, +1\}$
- ❖ Training data
 - ❖ $S = \{(x_i, y_i)\}$, m examples
- ❖ Hypothesis space

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Logistic Regression: Setup

- ❖ Training data

- ❖ $S = \{(x_i, y_i)\}$, m examples

- ❖ Hypothesis space

$$H = \{ h \mid h : X \rightarrow Y, h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- ❖ Learning Goal:

- ❖ Find an $h \in H$, such that $h(x) \approx P(y = 1 \mid x)$

- ❖ How to?

Maximum Likelihood

Which bag of words more likely generate:

aDaaa



Maximum Likelihood

Which bag of words more likely generate:

aDaaa

$$0.7 \times 0.1 \times 0.7 \times 0.7 \times 0.7 \\ = 2.401 \times 10^{-2}$$



$$0.2 \times 0.1 \times 0.2 \times 0.2 \times 0.2 \\ = 1.6 \times 10^{-4}$$



Drawing color cards from the envelope

- ❖ Let say we have several cards in the envelope
- ❖ Assume



$$P(\text{card} = \text{yellow}) = \theta$$

$$P(\text{card} = \text{purple}) = 1 - \theta$$

Drawing color cards from the envelope

- ❖ Sample with replacement n times
- ❖ k times we get yellow card, and n-k times, we get purple card
- ❖ The joint probability (likelihood)
$$C_K^N \theta^k (1 - \theta)^{n-k}$$
- ❖ What is the best θ making the joint probability maximal?

Drawing color cards from the envelope

$$\cancel{C_K} \theta^k (1 - \theta)^{n-k}$$

- ❖ Solving $\max_{\theta} \theta^k (1 - \theta)^{n-k}$
- ❖ Equivalently, we can solve

$$\max_{\theta} \log(\theta^k (1 - \theta)^{n-k})$$

$$\max_{\theta} k \log \theta + (n - k) \log(1 - \theta)$$

- ❖ At the optimum,

$$\frac{d(k \log \theta + (n - k) \log(1 - \theta))}{d\theta} = 0$$

The usual trick: Convert products to sums by taking log

Recall that this works only because log is an increasing function and the maximizer will not change

Drawing color cards from the envelope

$$\frac{d(k \log \theta + (n-k) \log(1-\theta))}{d\theta} = 0$$

$$\Rightarrow \frac{k}{\theta} - \frac{n-k}{1-\theta} = 0$$

For this simple problem, we have a closed-form solution.
We are not always lucky like this

$$\Rightarrow \theta(n - k) = (1 - \theta)k$$

$$\Rightarrow \theta = \frac{k}{n}$$



Maximum Likelihood Estimator (formal definition)

Likelihood function of parameters

Let X_1, \dots, X_N be **IID** (independent and identically distributed) with PDF $p(x|\theta)$ (also written as $p(x; \theta)$). The *likelihood function* is defined by $L(\theta)$,

$$L(\theta) = p(X_1, \dots, X_N; \theta). \quad = \prod_{i=1}^N p(X_i; \theta).$$

Notes The likelihood function is just the joint density of the data, except that we treat it as a function of the parameter θ .

Maximum Likelihood Estimation

Maximum Likelihood Estimator

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

Back to Logistic regression

Training data

$S = \{(x_i, y_i)\}$, m examples

Hypothesis space

$$H = \{ h \mid h : X \rightarrow Y, h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Find an $h \in H$ such that $h(x) \approx P(y = 1 \mid x)$

Back to Logistic regression

Training data

$S = \{(x_i, y_i)\}$, m examples

Hypothesis space

$$H = \{ h \mid h : X \rightarrow Y, h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Find an $h \in H$ such that $h(x) \approx P(y = 1 \mid x)$

How to? **Maximum Likelihood Estimator**

Likelihood function



$$P(\text{card} = \text{yellow}) = \theta \quad P(\text{card} = \text{purple}) = 1 - \theta$$

Find θ by maximizing $L(D; \theta)$

Logistic regression

Find w, b by maximizing $P(S; w, b)$

Likelihood function



$$P(\text{card} = \text{yellow}) = \theta \quad P(\text{card} = \text{purple}) = 1 - \theta$$

Find θ by maximizing $L(D; \theta)$

Logistic regression

Find w, b by maximizing $P(S; w, b)$

$$\operatorname{argmax}_{w,b} P(S; w, b) = \operatorname{argmax}_{w,b} \prod_{i=1}^m P(y_i | x_i; w, b)$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{w,b} P(S; w, b) = \operatorname{argmax}_{w,b} \prod_{i=1}^m P(y_i | x_i; w, b)$$

Equivalent to solve

$$\operatorname{argmax}_{w,b} \sum_{i=1}^m \log P(y_i | x_i; w, b)$$

Remember our assumption:

$$P(y = 1 | x; w, b) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-(w^T x + b))}$$

$$P(y = -1 | x; w, b) = 1 - \sigma(w^T x + b) = 1 - \frac{1}{1 + \exp(-(w^T x + b))}$$
$$= \frac{\exp(-(w^T x + b))}{1 + \exp(-(w^T x + b))} = \frac{1}{1 + \exp((w^T x + b))} = \sigma(-(w^T x + b))$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{w,b} P(S; w, b) = \operatorname{argmax}_{w,b} \prod_{i=1}^m P(y_i | x_i; w, b)$$

Equivalent to solve

$$\operatorname{argmax}_{w,b} \sum_{i=1}^m \log P(y_i | x_i; w, b)$$

Remember our assumption:

$$P(y = 1 | x; w, b) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-(w^T x + b))}$$

$$P(y_i | x_i; w, b) = \begin{cases} \sigma(w^T x_i + b) \\ \sigma(-(w^T x_i + b)) \end{cases} \Rightarrow P(y_i | x_i; w, b) = \sigma(y_i(w^T x_i + b))$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{w,b} P(S; w, b) = \operatorname{argmax}_{w,b} \prod_{i=1}^m P(y_i | x_i; w, b)$$

Equivalent to solve

$$\operatorname{argmax}_{w,b} \sum_{i=1}^m \log P(y_i | x_i; w, b)$$

Using $P(y_i | x_i; w, b) = \sigma(y_i(w^T x_i + b))$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$\begin{aligned} & \operatorname{argmax}_{w,b} \sum_{i=1}^m \log \sigma(y_i(w^T x_i + b)) \\ &= - \sum_{i=1}^m \log(1 + \exp(-y_i(w^T x_i + b))) \end{aligned}$$

How to Optimize the Loss?

How to minimizing the loss

- ❖ Optimization methods
 - ❖ Gradient Descent
 - ❖ Stochastic Gradient Descent
 - ❖ Analytic solution
- ❖ ...many other approaches

How to solve it?

$$\operatorname{argmax}_{w,b} - \sum_{i=1}^m \log(1 + \exp(-y_i(w^T x_i + b)))$$

There is no closed-form solution

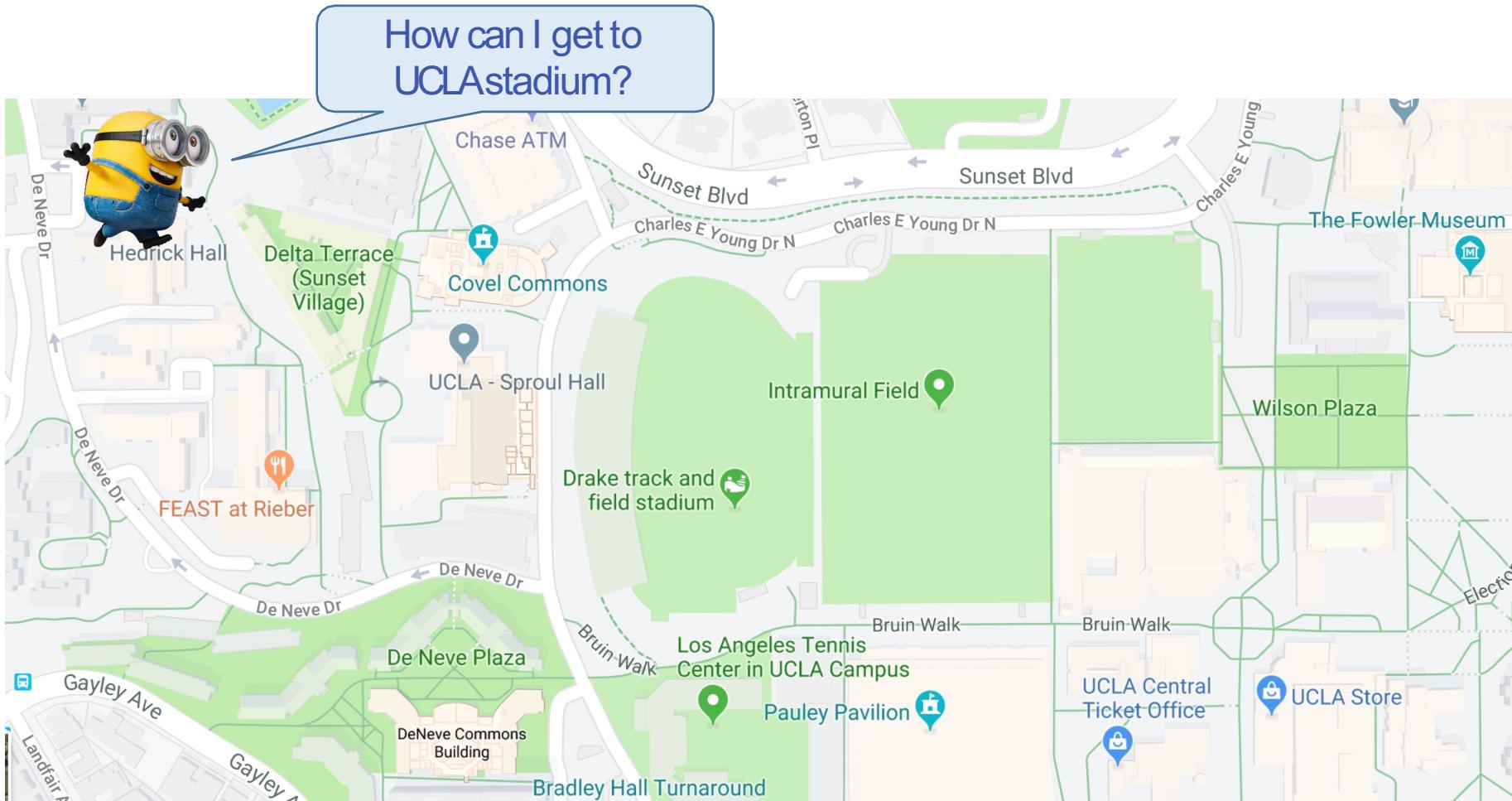
Max f(x) is equivalent to min - f(x)
=>only need to consider minimization problems.

One way to solve it is by gradient descent

Gradient Descent

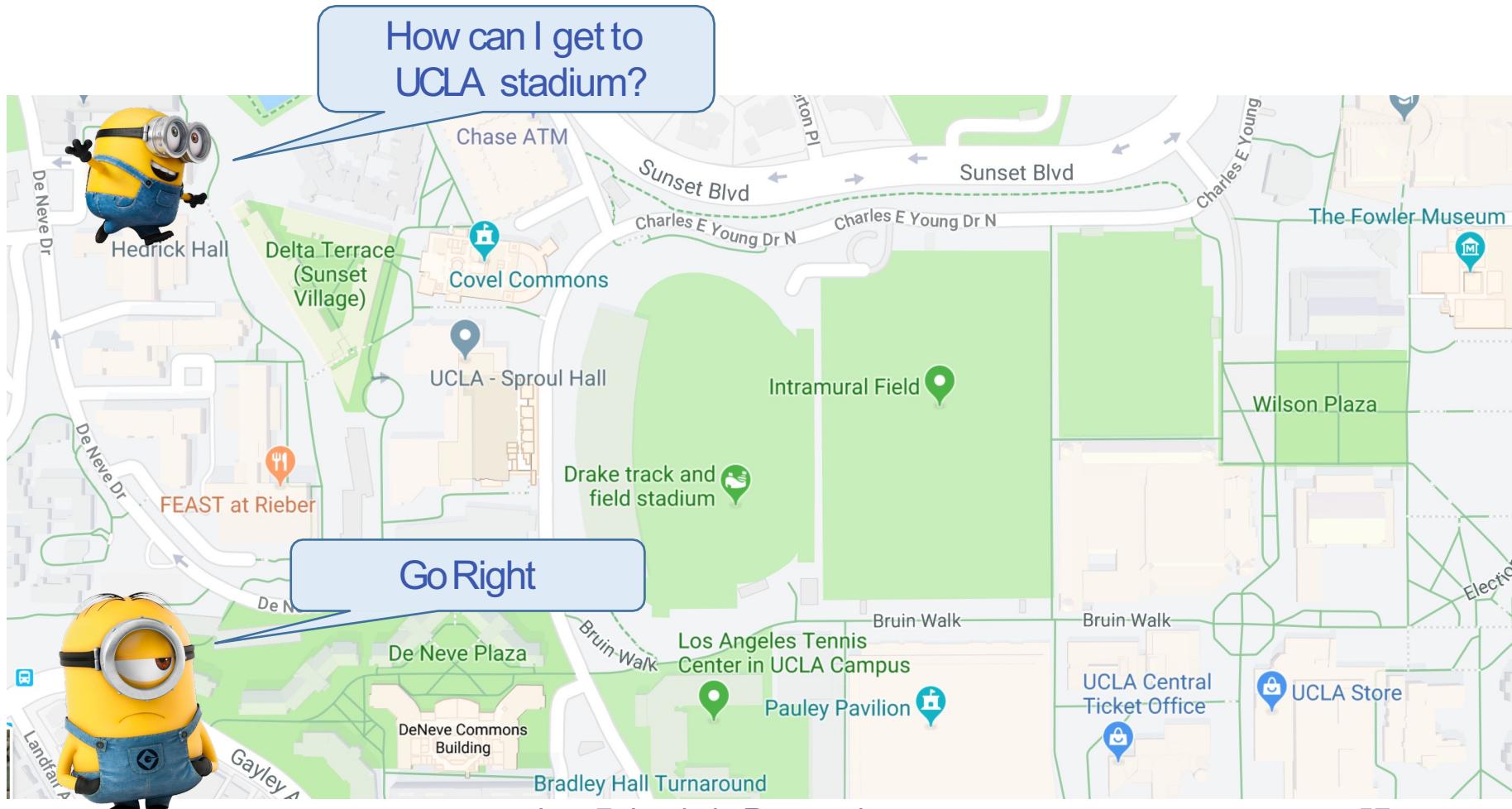
Intuition

Asking direction



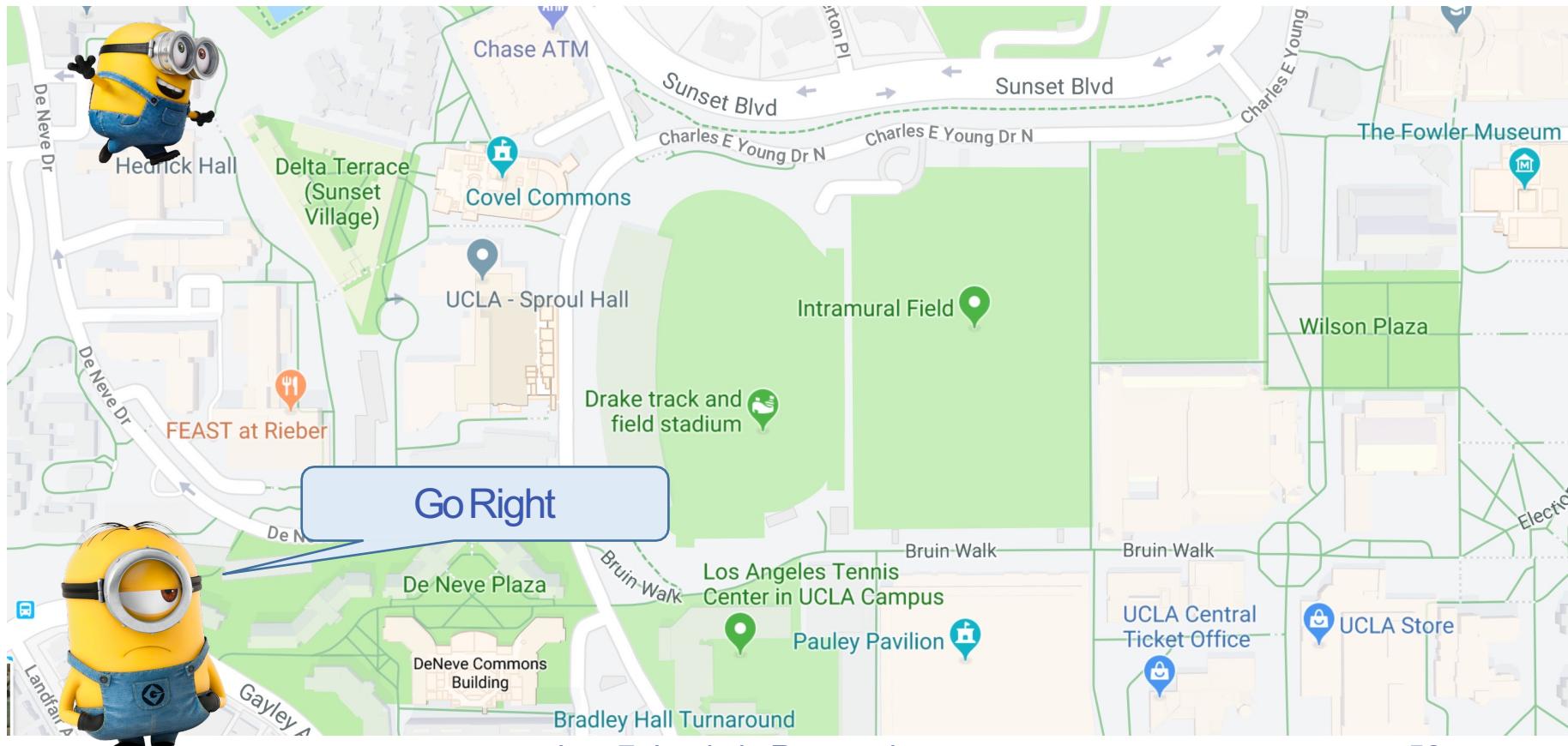
Intuition

Asking direction



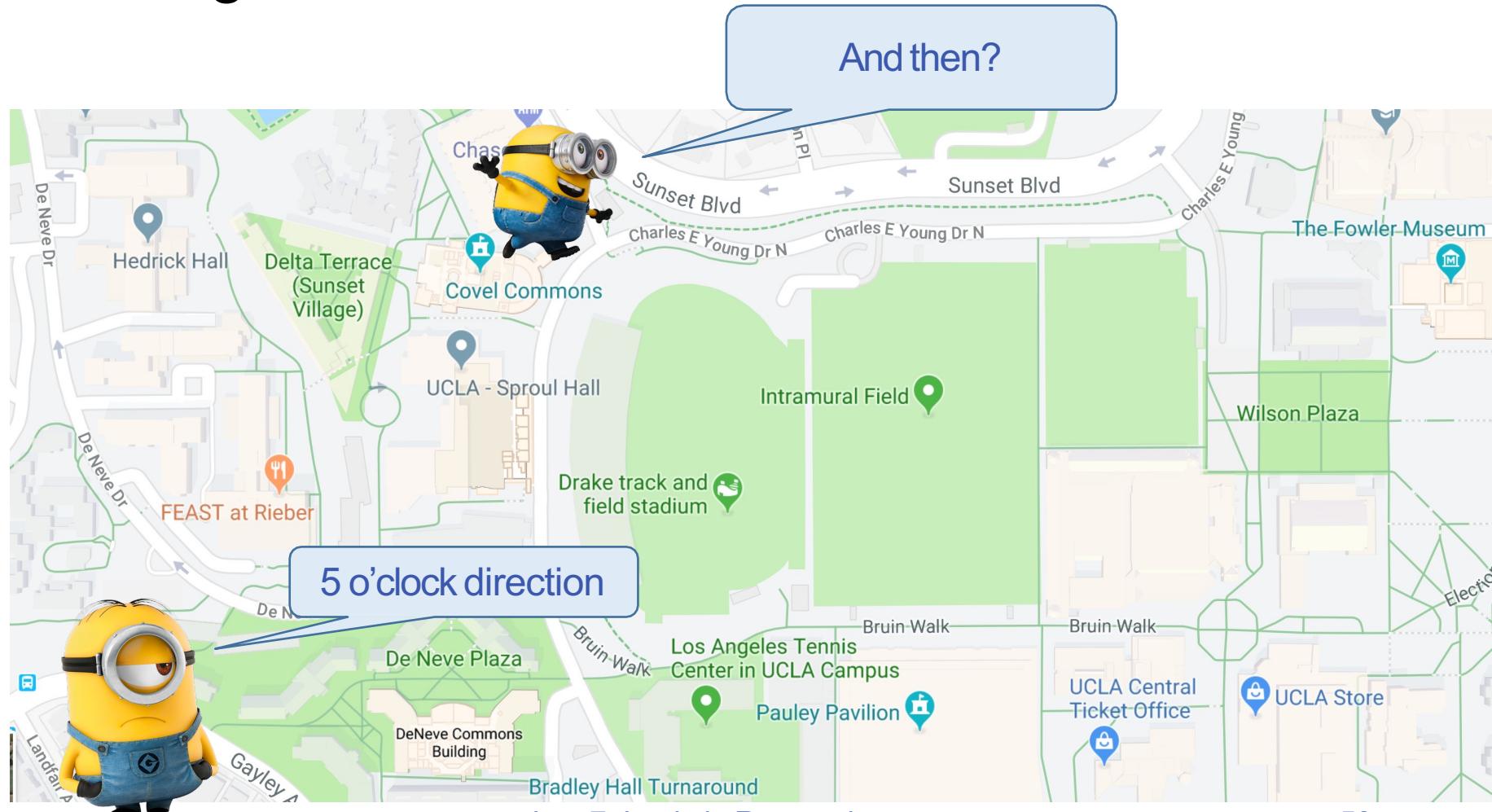
Intuition

Asking direction



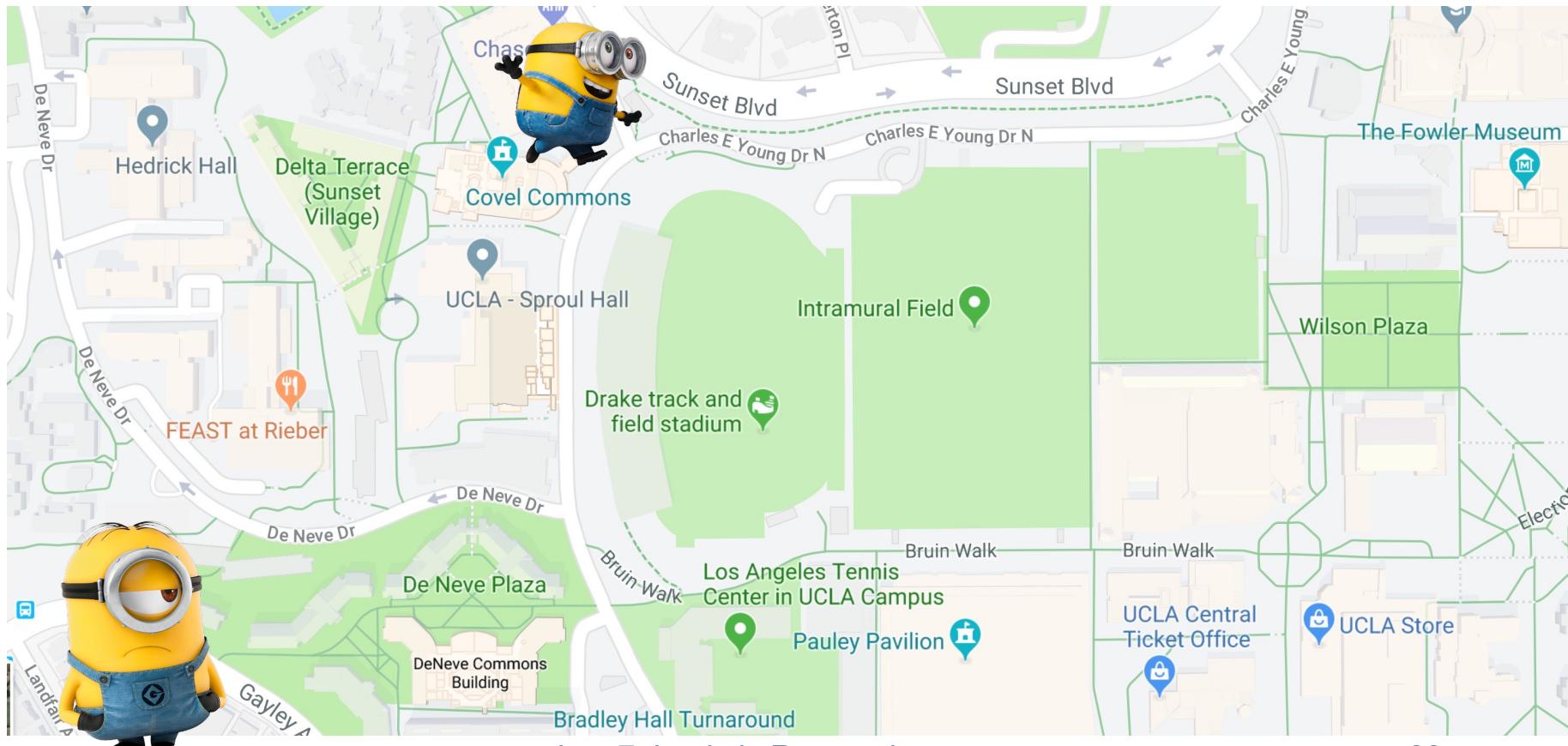
Intuition

Asking direction



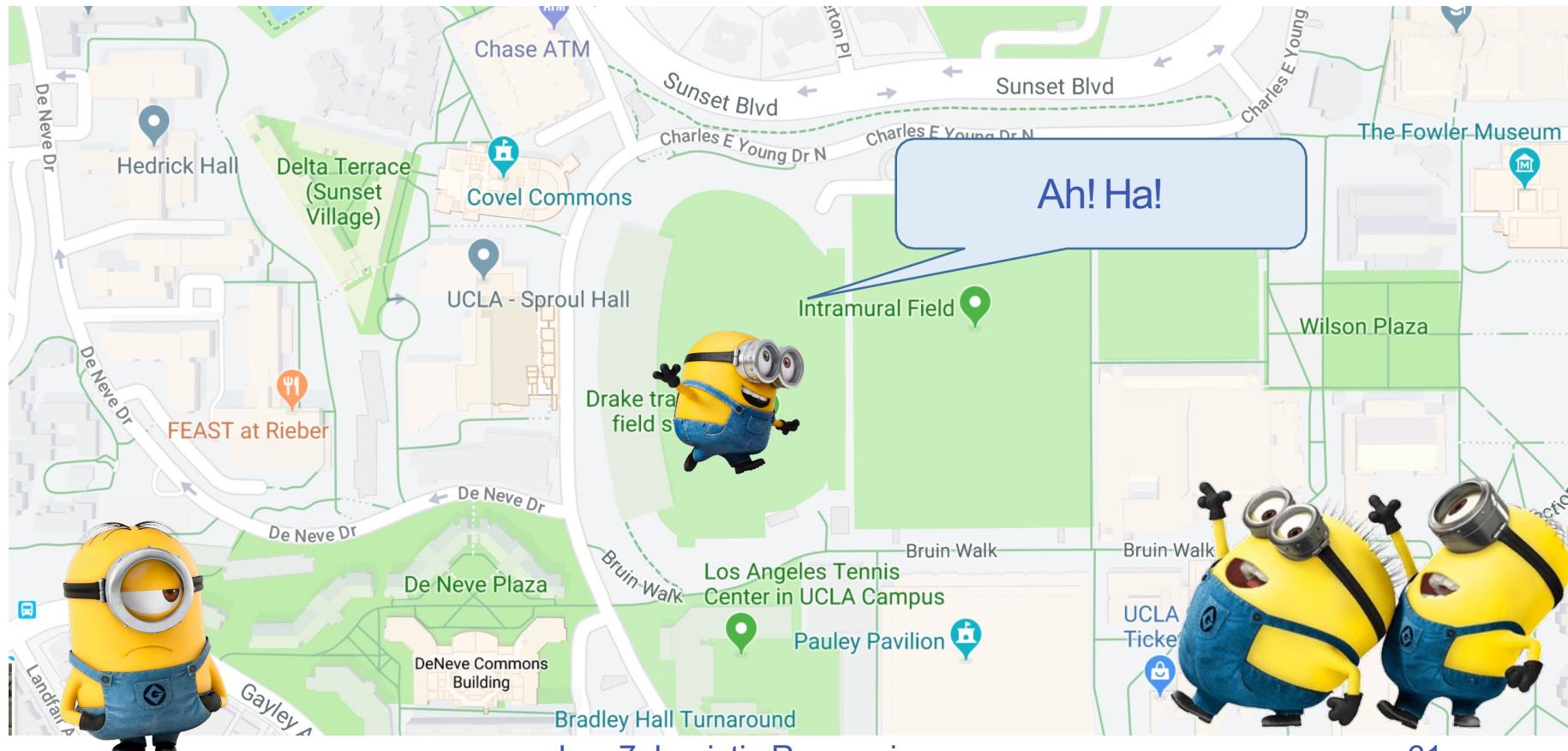
Intuition

Asking direction



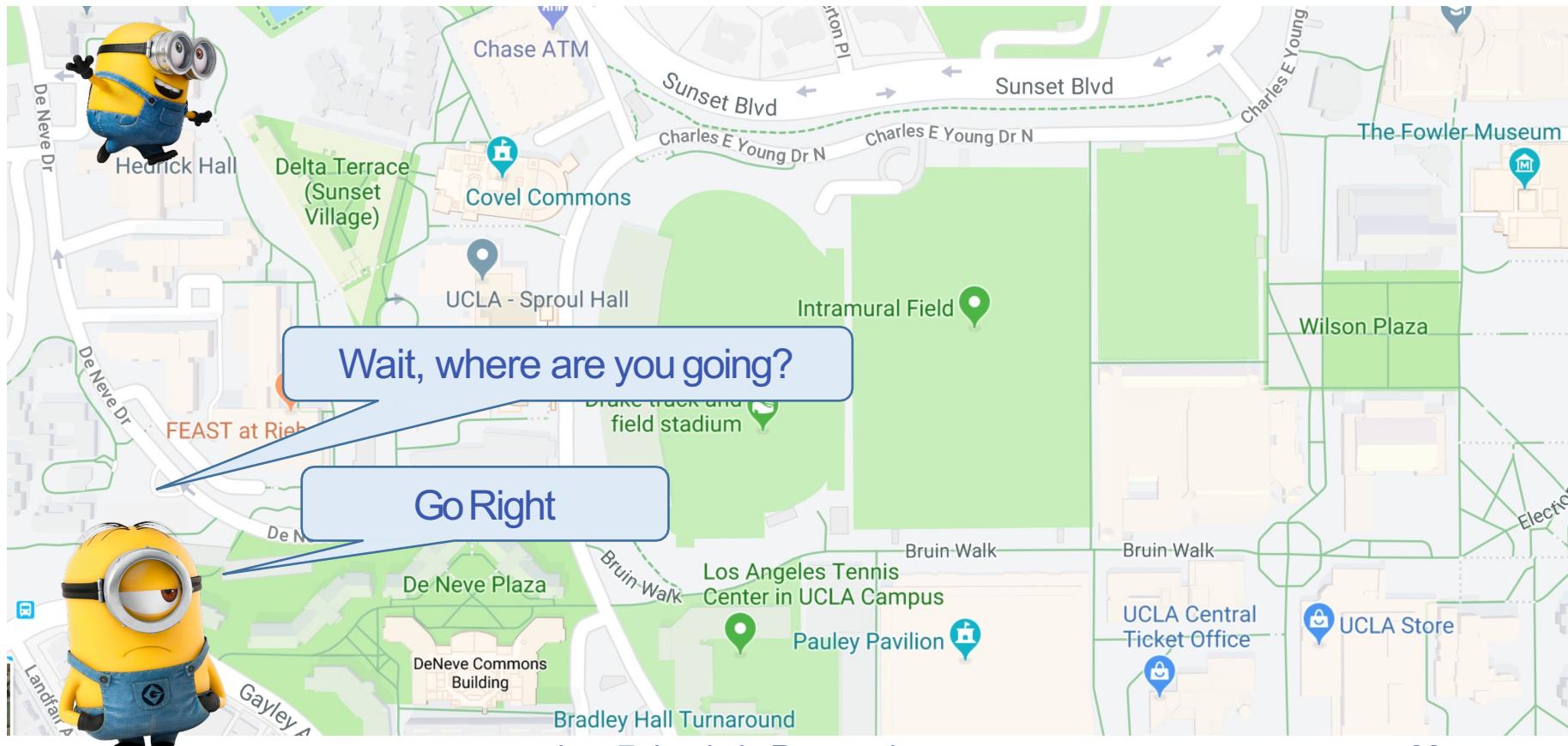
Intuition

Asking direction



Intuition

What may go wrong? Incorrect Step-size

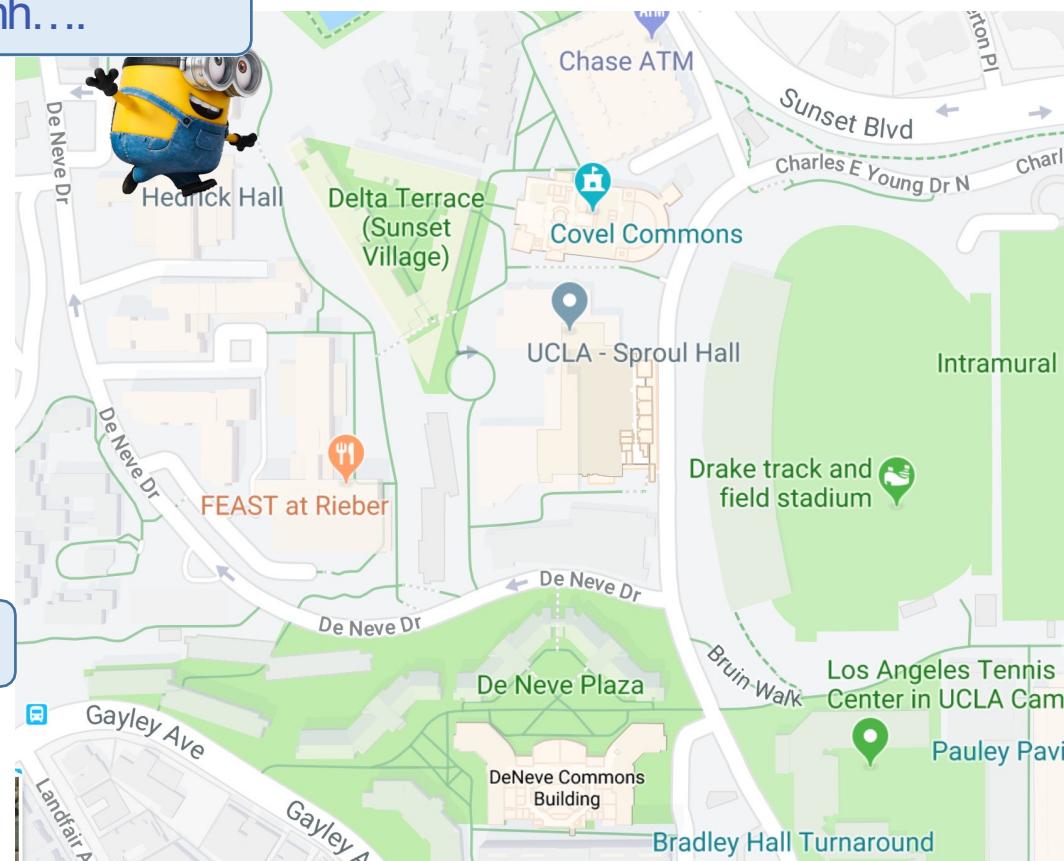


Intuition

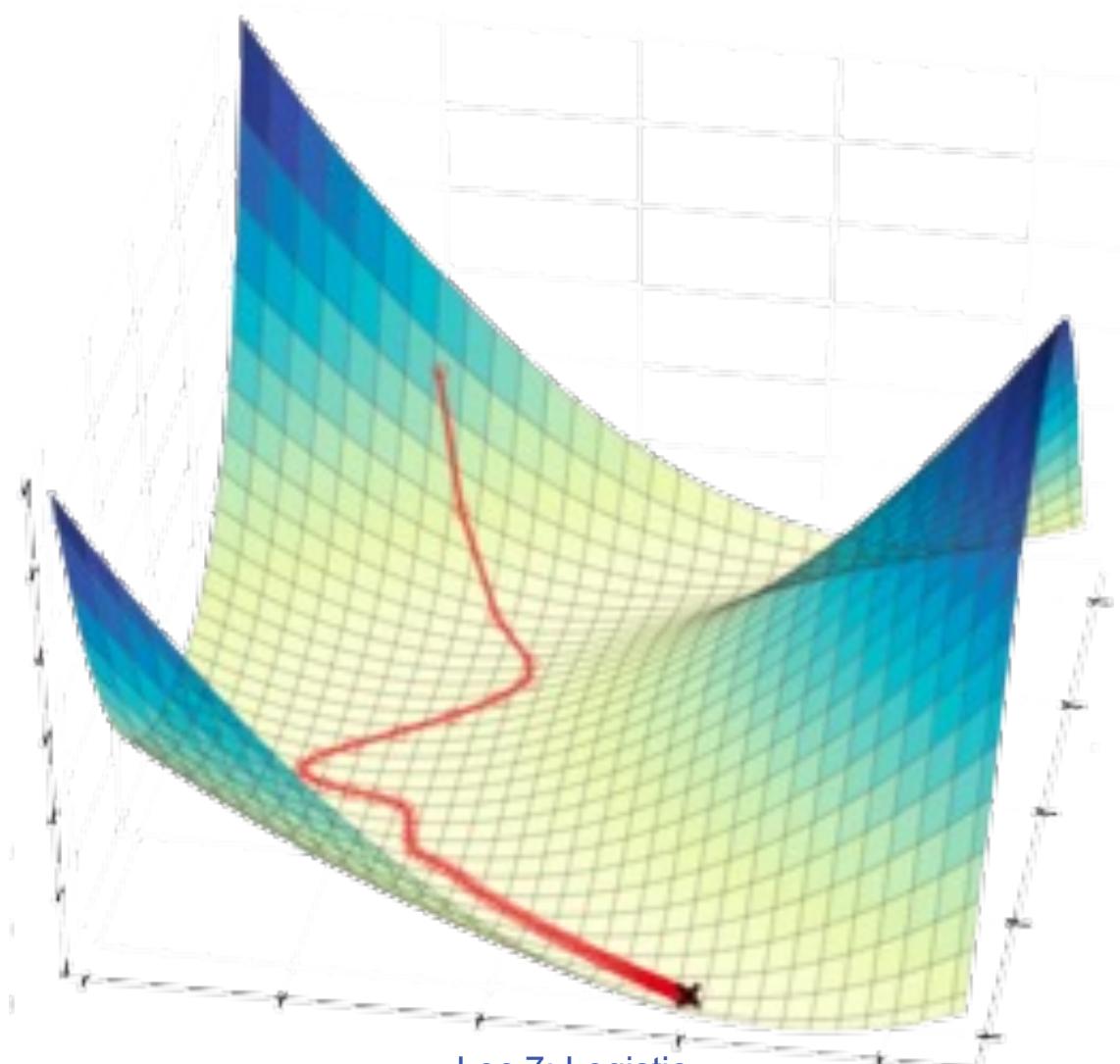
What may go wrong? Incorrect Direction



AHHhhhhhhh....



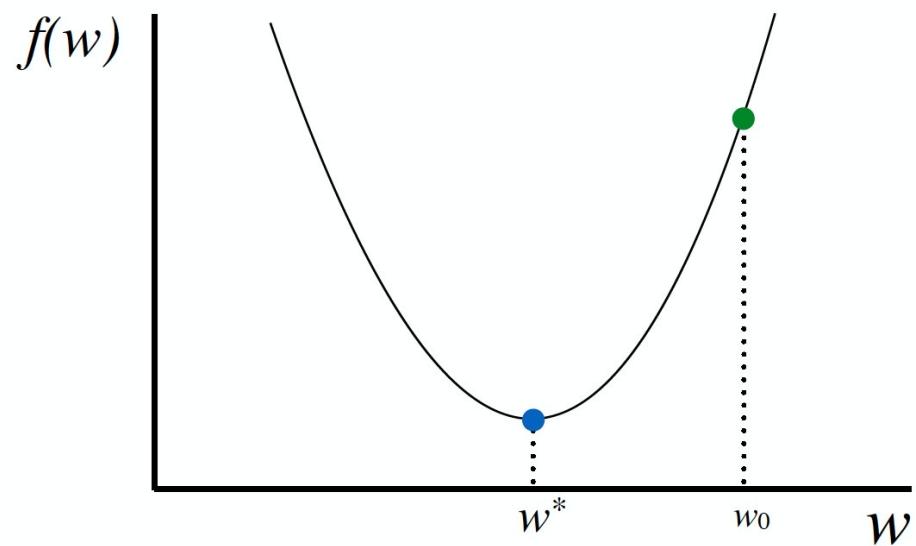
Gradient Descent



Lec 7: Logistic
Regression

Gradient descent

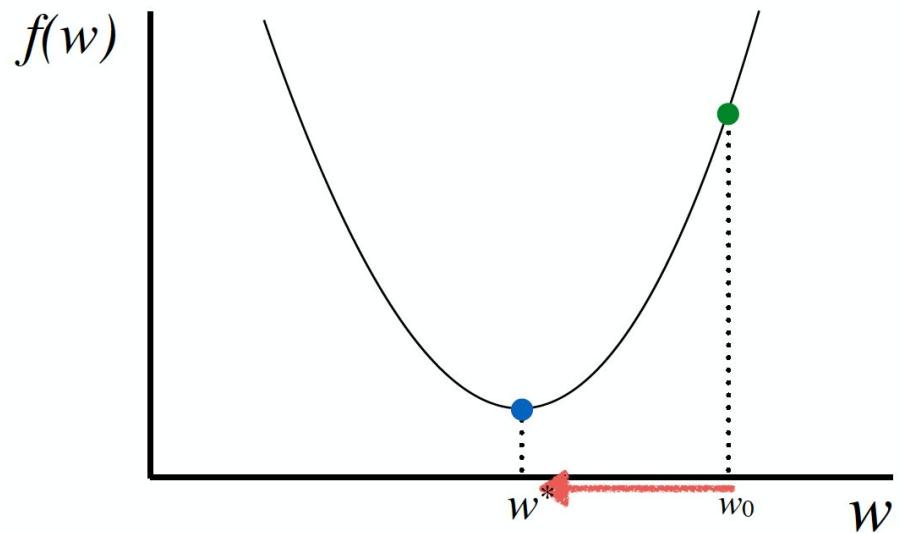
Start at a random point



Gradient descent

Start at a random point

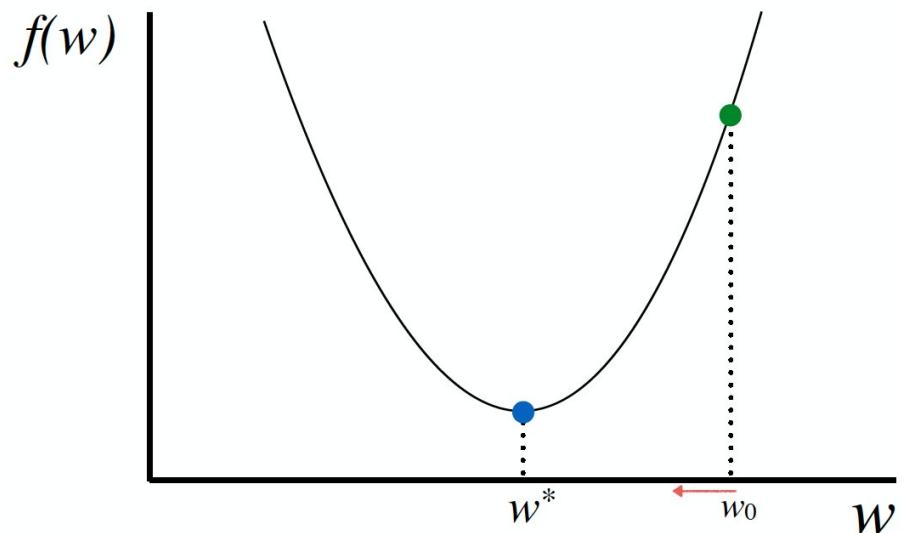
Determine a descent direction



Gradient descent

Start at a random point

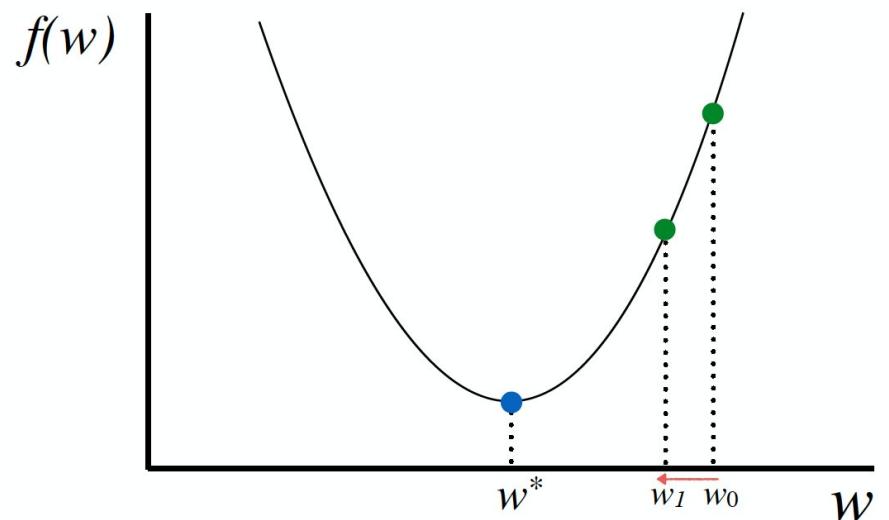
Determine a descent direction
Choose a step size



Gradient descent

Start at a random point

Determine a descent direction
Choose a step size
Update



Gradient descent

Start at a random point

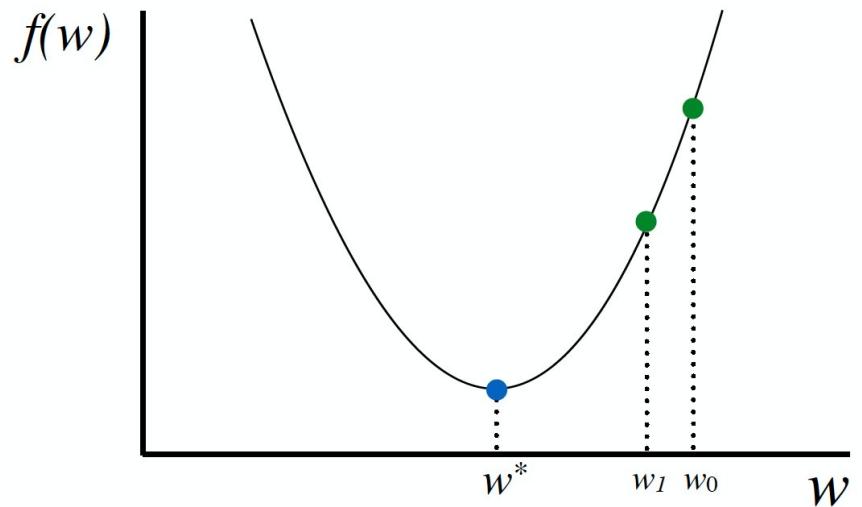
Repeat

Determine a descent direction

Choose a step size

Update

Until stopping criterion is satisfied



Gradient descent

Start at a random point

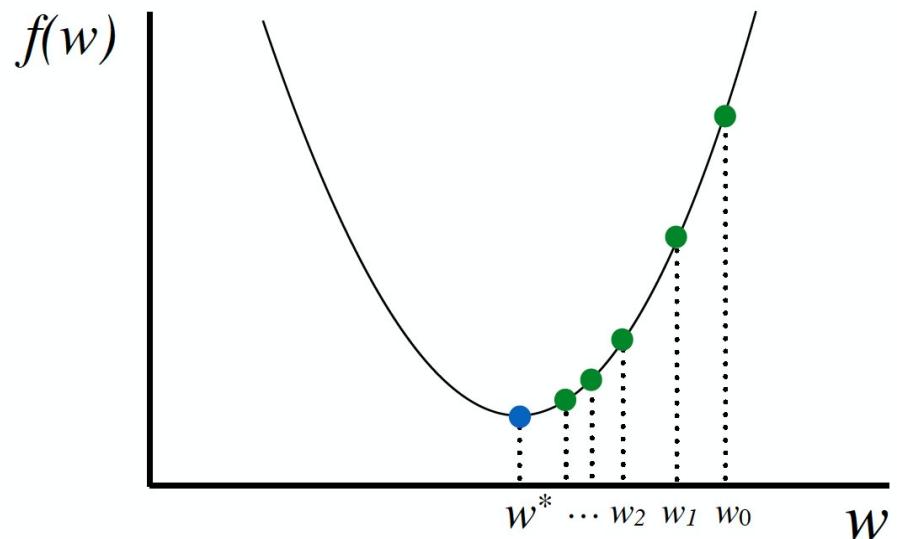
Repeat

Determine a descent direction

Choose a step size

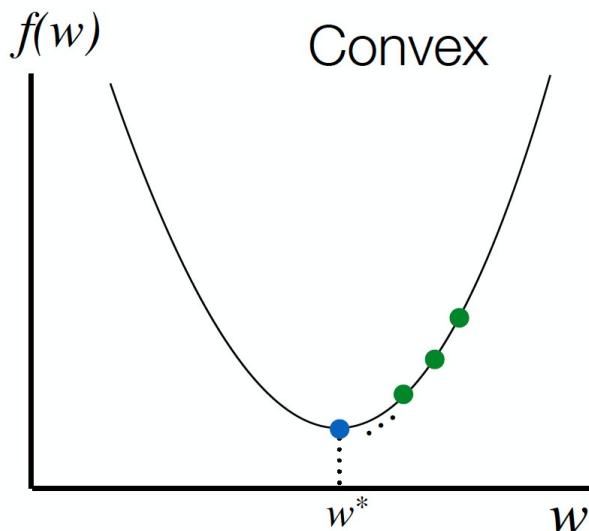
Update

Until stopping criterion is satisfied

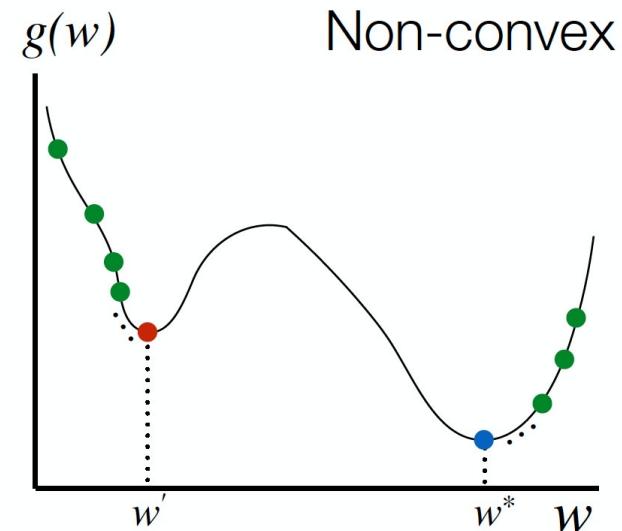


Where will we converge?

If function is convex, it converges to the global optimum (need proper choice of step-size)



Any local minimum is a global minimum



Multiple local minima may exist

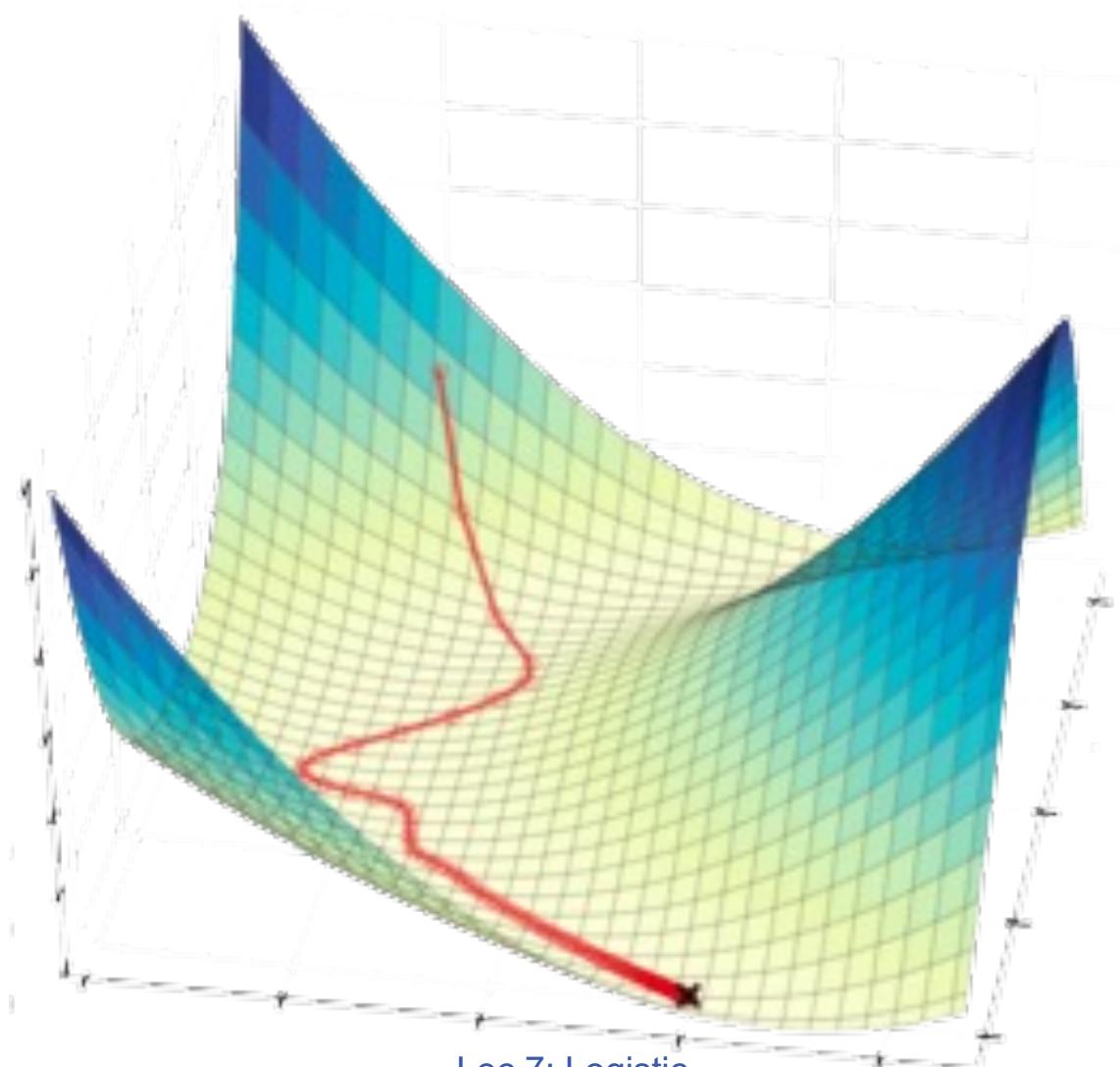
Gradient Descent

Algorithm 1 Gradient Descent (J)

- 1: $t \leftarrow 0$
 - 2: Initialize $\theta^{(0)}$
 - 3: **repeat**
 - 4: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla J(\theta^{(t)})$
 - 5: $t \leftarrow t + 1$
 - 6: **until** convergence
 - 7: Return final value of θ
-

Need to compute the gradient for the negative log likelihood

Gradient Descent



Lec 7: Logistic
Regression

Example

$$\min f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$$

We compute the gradients

$$\frac{\partial f}{\partial \theta_1} = 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1$$

$$\frac{\partial f}{\partial \theta_2} = -(\theta_1^2 - \theta_2)$$

Example $\min f(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$

❖ Use the following iterative procedure for gradient descent

- ① Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
- ② do

Type equation here.

$$\nabla f(\theta) = \begin{bmatrix} 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \\ -(\theta_1^2 - \theta_2) \end{bmatrix}$$

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2(\theta_1^{(t)})^2 - \theta_2^{(t)} \right] \theta_1^{(t)} + \theta_1^{(t)} - 1$$

$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-(\theta_1^{(t)})^2 - \theta_2^{(t)} \right]$$

$$t \leftarrow t + 1$$

- ③ until $f(\theta^{(t)})$ does not change much

Remarks

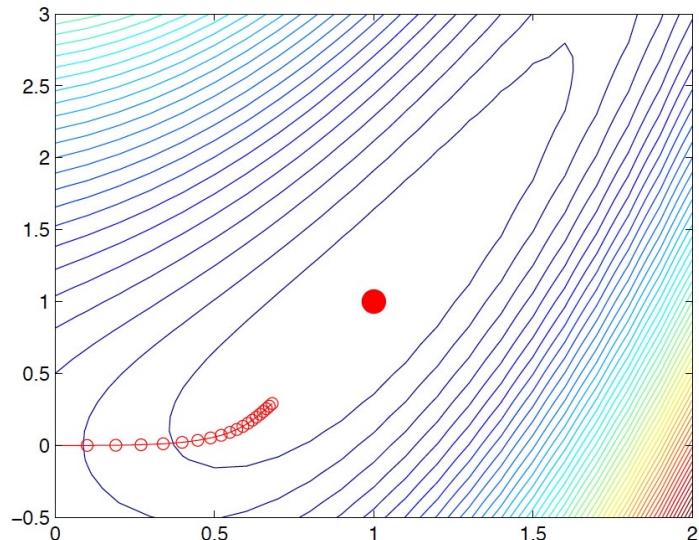
- ❖ η is often called step size or learning rate -- how far our update will go along the the direction of the negative gradient
- ❖ With a **suitable** choice of η , the iterative procedure converges to a stationary point where

$$\frac{\partial f}{\partial \theta} = 0$$

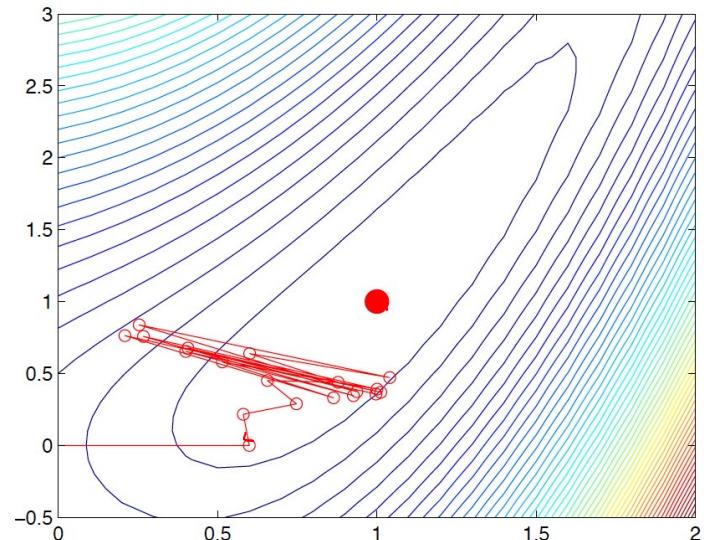
- ❖ A stationary point is only necessary for being the minimum

Choosing the right η is important

small η is too slow?

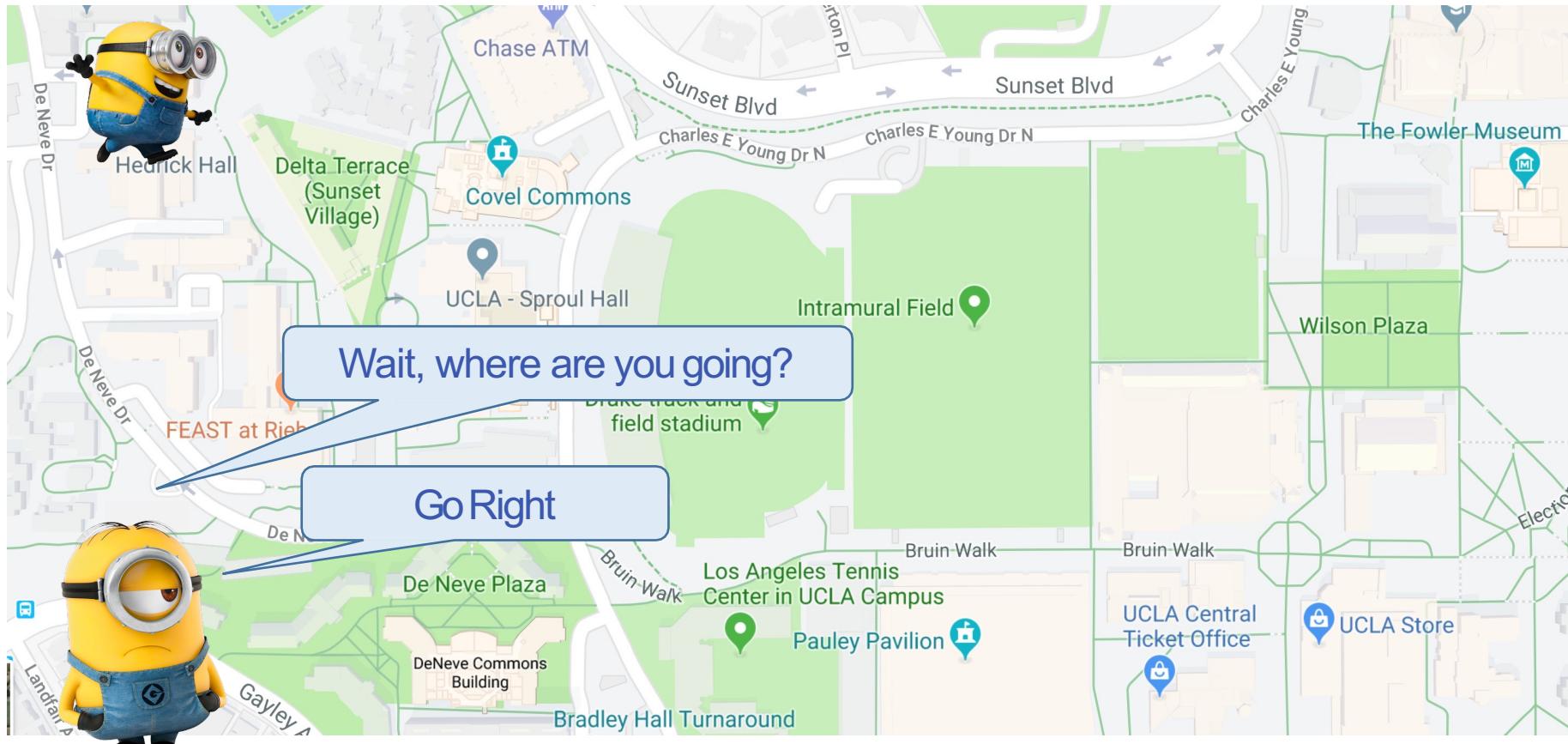


large η is too unstable?



Intuition

What may go wrong? Incorrect Step-size



Iterative optimization

Algorithm 2 Gradient Descent (J)

- 1: $t \leftarrow 0$
 - 2: Initialize $\theta^{(0)}$
 - 3: **repeat**
 - 4: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \boxed{\eta \nabla J(\theta^{(t)})}$ **How to compute the gradient?**
 - 5: $t \leftarrow t + 1$
 - 6: **until** convergence
 - 7: Return final value of θ
-

Need to compute the gradient for the linear regression cost function (residual sum of squares RSS)

Stochastic Gradient Descent

Incremental/Stochastic gradient descent

Repeat for each example (\mathbf{x}_i, y_i)

Use this example to calculate the
gradient and update the model

Contrast with *batch gradient descent* which
makes one update to the weight vector for
every pass over the data

Stochastic gradient descent

If $f(w) = \frac{1}{|D|} \sum_i^{|D|} f_i(w)$

$$\nabla f(w) = \frac{1}{|D|} \sum_i \nabla f_i(w) = E_{i \sim D} \nabla f_i(w)$$

- ❖ Approximate the true gradient by a gradient at a single example at a time

Repeat until converge:

Randomly pick one sample (x_i, y_i)

Update $w \leftarrow w - \eta \nabla f_i(w)$

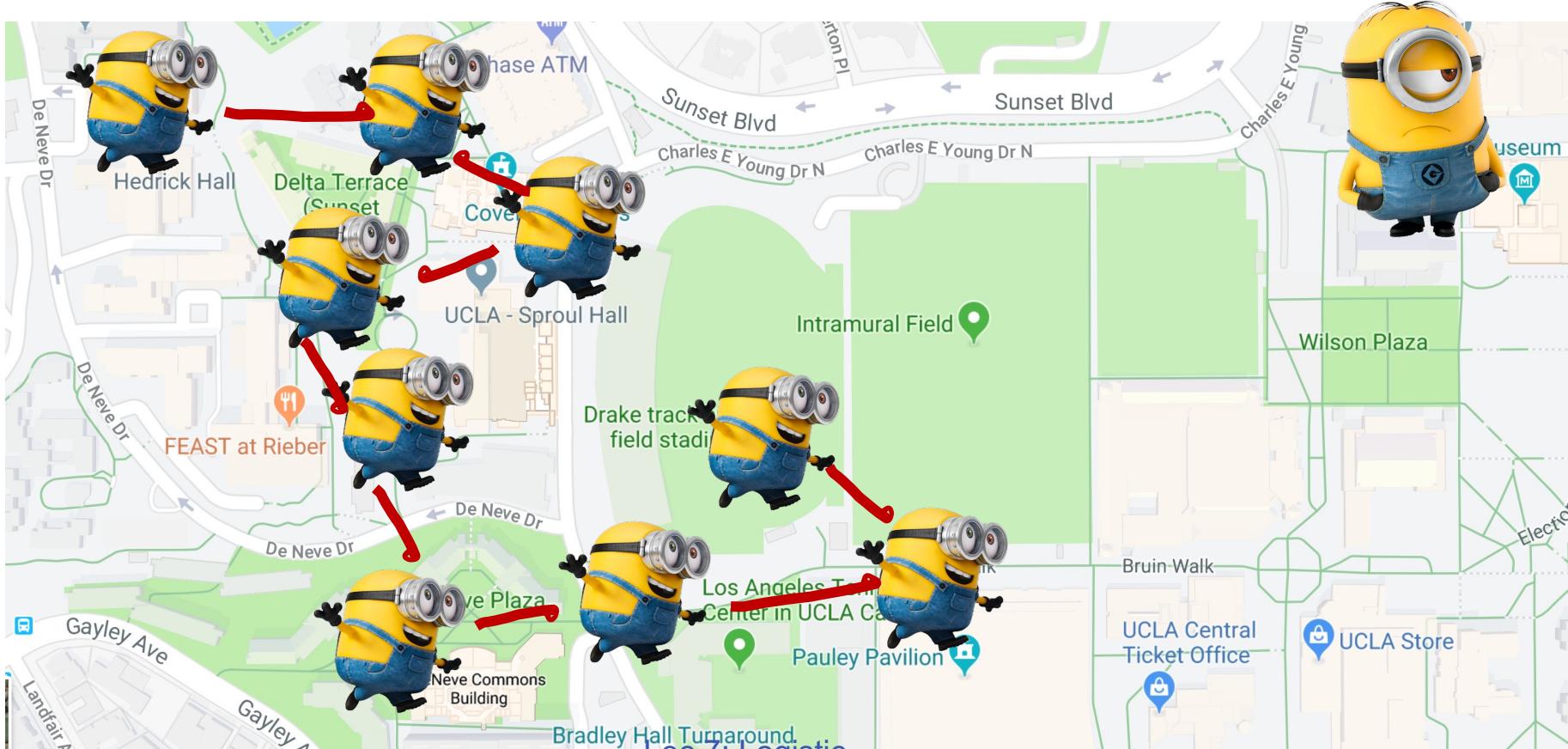
Intuition

Asking direction. Gradient descent:
compute gradient of all instances.



Intuition

Asking direction. Stochastic Gradient descent:
compute approximate gradient by one instance



Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
3. For (x, y) in \mathcal{D} :
4. Update $w \leftarrow w - \eta \nabla f(w)$
5. Return w

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For epoch 1 ... T :
 3. For (x, y) in \mathcal{D} :
 4. if $y(w^\top x) < 0$
 5. $w \leftarrow w + \eta yx$
 6. Return w

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch 1 ... T :
3. For (x, y) in \mathcal{D} :
4. if $y(w^T x) < 0$
5. $w \leftarrow w + \eta yx$
6. Return w

Prediction: y^{test}

Perceptron effectively minimizing:

$$\sum_i \max(0, 1 - y_i(w^T x_i))$$

Summary

- ❖ Maximum Likelihood Estimation
- ❖ Gradient Descent
- ❖ Stochastic Gradient Descent
- ❖ We will see more examples in later lectures

Lecture 8: Neural Network & Deep Learning Fall 2022

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcements

- ❖ Quiz 2 is due today
- ❖ Hw 1 is due next Tue
 - ❖ The definition of F1 score will be covered today
- ❖ Midterm postpones to 11/1?
- ❖ The practice exam will be posted

What you will learn today

- ❖ Optimization
 - ❖ Gradient descent
 - ❖ Stochastic gradient descent (SGD)
- ❖ Evaluation Metrics
- ❖ Neural network / Deep learning
 - ❖ Non-linear classifier
 - ❖ Feed-forward neural network
 - ❖ Deep learning architecture

Gradient Descent

Example $\min f(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$

❖ Use the following iterative procedure for gradient descent

- ➊ Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
- ➋ do

Type equation here.

$$\nabla f(\theta) = \begin{bmatrix} 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \\ -(\theta_1^2 - \theta_2) \end{bmatrix}$$

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2(\theta_1^{(t)})^2 - \theta_2^{(t)} \right] \theta_1^{(t)} + \theta_1^{(t)} - 1$$

$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-(\theta_1^{(t)})^2 - \theta_2^{(t)} \right]$$

$$t \leftarrow t + 1$$

- ➌ until $f(\theta^{(t)})$ does not change much

Remarks

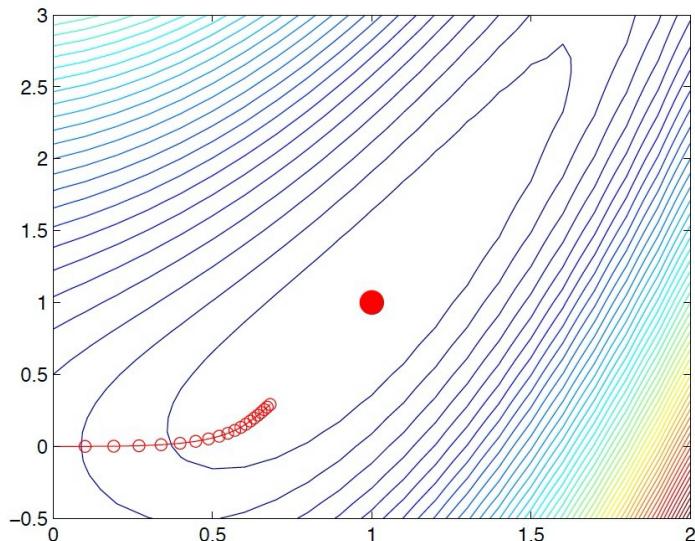
- ❖ η is often called step size or learning rate -- how far our update will go along the the direction of the negative gradient
- ❖ With a **suitable** choice of η , the iterative procedure converges to a stationary point where

$$\frac{\partial f}{\partial \theta} = 0$$

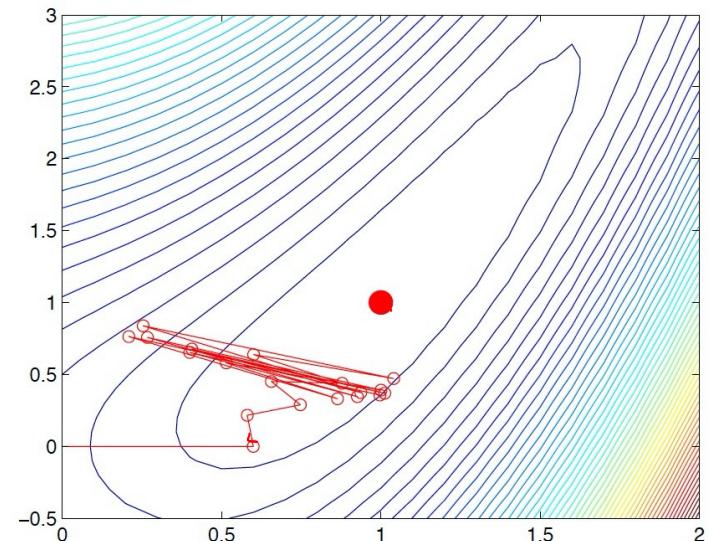
- ❖ A stationary point is only necessary for being the minimum

Choosing the right learning rate (η) is important

small η is too slow?



large η is too unstable?



Recap: Logistic Regression

- ❖ Training data: $S = \{(x_i, y_i)\}$, m examples
- ❖ Hypothesis space:

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

i.e., model $P(Y|X)$ by $\sigma(w^T x + b)$

- ❖ How to find the best $h \in H$: maximum log-likelihood

$$\arg \max - \sum_{i=1}^m \log(1 + \exp(-y_i(w^T x_i + b)))$$

Gradient Descent for Logistic Regression

- ❖ Maximum log-likelihood

$$\arg \max - \sum_{i=1}^m \log(1 + \exp(-y_i(w^T x_i + b)))$$

- ❖ Equivalent to the following minimization problem

$$\arg \min \underbrace{\sum_{i=1}^m \log(1 + \exp(-y_i(w^T x_i + b)))}_{L(w, b)}$$

- ❖ Gradient of $L(w, b)$

$$\nabla L(w, b) = \sum_{i=1}^m \nabla \log(1 + \exp(-y_i(w^T x_i + b)))$$

Recap: Gradient

- ❖ Let z to be a n -dimensional vector of variables, $f(z)$ is a function of z

$$\nabla f(z) = \begin{bmatrix} \partial f(z)/\partial z_1 \\ \partial f(z)/\partial z_2 \\ \vdots \\ \partial f(z)/\partial z_{n-1} \\ \partial f(z)/\partial z_n \end{bmatrix}$$

Exercise

- ❖ Let $z = [z_1, z_2, z_3]^T$ to be a 3-dimensional vector of variables, $a = [3, 2, 4]^T$

$$\begin{aligned}f(z) &= \log(a^T z) \\&= \log(3z_1 + 2z_2 + 4z_3)\end{aligned}$$

$$\nabla f(z) = \begin{bmatrix} \partial f(z)/\partial z_1 \\ \partial f(z)/\partial z_2 \\ \vdots \\ \partial f(z)/\partial z_{n-1} \\ \partial f(z)/\partial z_n \end{bmatrix}$$

- ❖ $\nabla f(z) = \begin{bmatrix} \partial f(z)/\partial z_1 \\ \partial f(z)/\partial z_2 \\ \partial f(z)/\partial z_3 \end{bmatrix} = ?$

Exercise

- ❖ Let $z = [z_1, z_2, z_3]^T$ to be a 3-dimensional vector of variables, $a = [3, 2, 4]^T$

$$\begin{aligned}f(z) &= \log(a^T z) \\&= \log(3z_1 + 2z_2 + 4z_3)\end{aligned}$$

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f(z)}{\partial z_1} \\ \frac{\partial f(z)}{\partial z_2} \\ \vdots \\ \frac{\partial f(z)}{\partial z_{n-1}} \\ \frac{\partial f(z)}{\partial z_n} \end{bmatrix}$$

$$\begin{aligned}\nabla f(z) &= \begin{bmatrix} \frac{\partial f(z)}{\partial z_1} \\ \frac{\partial f(z)}{\partial z_2} \\ \frac{\partial f(z)}{\partial z_3} \end{bmatrix} = \begin{bmatrix} \frac{3}{3z_1+2z_2+4z_3} \\ \frac{2}{3z_1+2z_2+4z_3} \\ \frac{4}{3z_1+2z_2+4z_3} \end{bmatrix} \\&= \frac{1}{3z_1+2z_2+4z_3} \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix} = \frac{1}{a^T z} a\end{aligned}$$

Gradient of $L(w, b)$

$$\nabla L(w, b) = \sum_{i=1}^m \nabla \underbrace{\log(1 + \exp(-y_i(w^T x_i + b)))}_{\nabla \log \frac{1}{\sigma(y_i(w^T x_i + b))}}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$\begin{aligned}\diamond \quad \nabla \log \frac{1}{\sigma(z)} &= \nabla \log(1 + \exp(-z)) = -\frac{\exp(-z)}{1 + \exp(-z)} \\ &= -\frac{1 + \exp(-z) - 1}{1 + \exp(-z)} = -1 + \frac{1}{1 + \exp(-z)} \\ &= \sigma(z) - 1\end{aligned}$$

Gradient of $L(w, b)$

$$\nabla L(w, b) = \sum_{i=1}^m \nabla \underbrace{\log(1 + \exp(-y_i(w^T x_i + b)))}_{\sigma(z)} \quad \text{where } \sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \qquad \qquad \nabla \log \frac{1}{\sigma(y_i(w^T x_i + b))}$$

❖ Using $\nabla \log \frac{1}{\sigma(z)} = \sigma(z) - 1$

Partial gradient w.r.t w

$$\begin{aligned}\nabla_w L(w, b) &= \sum_{i=1}^m \nabla_w \log \frac{1}{\sigma(y_i(w^T x_i + b))} \\ &= \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1) y_i x_i\end{aligned}$$

$$\nabla_b L(w, b) = \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1) y_i$$

Gradient descent for logistic regression

Given a training data set $S = \{(x_i, y_i)\}, i = 1 \dots m$

1. Initialize w (e.g., $w \leftarrow 0 \in R^n$)
2. For epoch $1 \dots T$:
 Loop over instance to compute the summation
3. Compute $\nabla_w L(w, b)$ and $\nabla_b L(w, b)$
$$\nabla_w L(w, b) = \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1)y_i x_i$$
$$\nabla_b L(w, b) = \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1)y_i$$
4. Update w and b
$$w \leftarrow w - \eta \nabla_w L(w, b)$$
$$b \leftarrow b - \eta \nabla_b L(w, b)$$
5. Return w and b

Remark

$$\nabla_w L(w, b) = \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1)y_i x_i$$
$$\nabla_b L(w, b) = \sum_{i=1}^m (\sigma(y_i(w^T x_i + b)) - 1)y_i$$

- ❖ Need to compute $(\sigma(y_i(w^T x_i + b)) - 1)$ for every data point (x_i, y_i)
- ❖ Gradient descent usually needs many iterations to converge
- ❖ When size of data (m) is large, computing $\nabla L(w, b)$ is expensive

Stochastic Gradient Descent

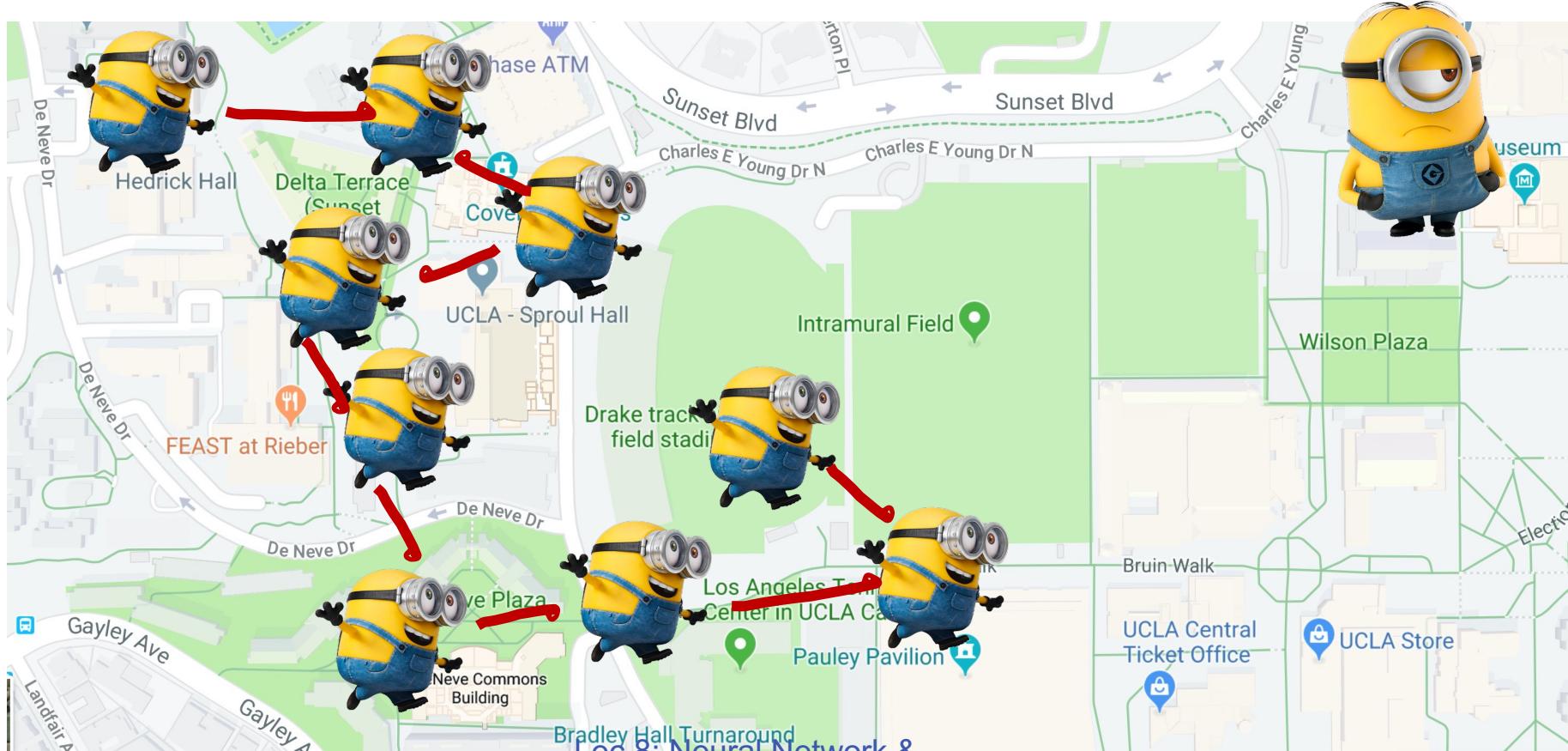
Intuition

Asking direction. Gradient descent:
compute gradient of all instances.



Intuition

Asking direction. Stochastic Gradient descent:
compute approximate gradient by one instance



Incremental/Stochastic gradient descent

Repeat for each example (\mathbf{x}_i, y_i)

Use this example to calculate approximate the gradient and update the model

Contrast with *batch gradient descent* which makes one update to the weight vector for every pass over the data

Recap: Gradient Descent

- ❖ $\nabla_w L(w, b) = \sum_{i=1}^m (\underbrace{\sigma(y_i(w^T x_i + b)) - 1}_{\nabla L_i(w, b)} y_i x_i)$

- ❖ Gradient descent update:

$$w \leftarrow w - \eta \sum_{i=1}^m \nabla_w L_i(w, b)$$

- ❖ Alternative way of gradient update

For $i = 1 \dots m$

$$w \leftarrow w - \eta \nabla_w L_i(w, b)$$

Stochastic Gradient Descent

- ❖ $\nabla_w L(w, b) = \sum_{i=1}^m \nabla_w L_i(w, b) = m \frac{\sum_{i=1}^m \nabla L_i(w, b)}{m} = m \underbrace{\text{avg}(\nabla_w L_i(w, b))}_{\text{Average } L_i(w, b) \text{ over instances}}$
- ❖ $\text{avg}(\nabla_w L_i(w, b)) = E_{(x_i, y_i) \sim S} [\nabla_w L_i(w, b)]$ Average $L_i(w, b)$ over instances
- ❖ Gradient descent update:
 $w \leftarrow w - \eta \sum_{i=1}^m \nabla_w L_i(w, b)$
- ❖ Stochastic gradient descent
 - ❖ Repeat until converge
 - Sample a data point (x_i, y_i) from S
 - $w \leftarrow w - \eta' \nabla_w L_i(w, b)$

Expectation of gradient
 $L_i(w, b)$ over dataset S

Stochastic Gradient descent for logistic regression

Given a training data set $S = \{(x_i, y_i)\}, i = 1 \dots m$

1. Initialize w (e.g., $w \leftarrow 0 \in R^n$)
2. For epoch 1 ... T :
3. Sample a data point (x_i, y_i) from S
4. Compute $\nabla_w L_i(w, b)$ and $\nabla_b L_i(w, b)$
$$\nabla_w L_i(w, b) = (\sigma(y_i(w^T x_i + b)) - 1)y_i x_i$$
$$\nabla_b L_i(w, b) = (\sigma(y_i(w^T x_i + b)) - 1)y_i$$
5. Update w and b
$$w \leftarrow w - \eta \nabla_w L_i(w, b)$$
$$b \leftarrow b - \eta \nabla_b L_i(w, b)$$
6. Return w and b

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch 1 ... T :
3. For (x, y) in \mathcal{D} :
4. if $y(w^\top x) < 0$
5. $w \leftarrow w + \eta yx$
6. Return w

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
3. For (\mathbf{x}, y) in \mathcal{D} :
4. if $y(\mathbf{w}^\top \mathbf{x}) < 0$
5. $\mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}$
6. Return \mathbf{w}

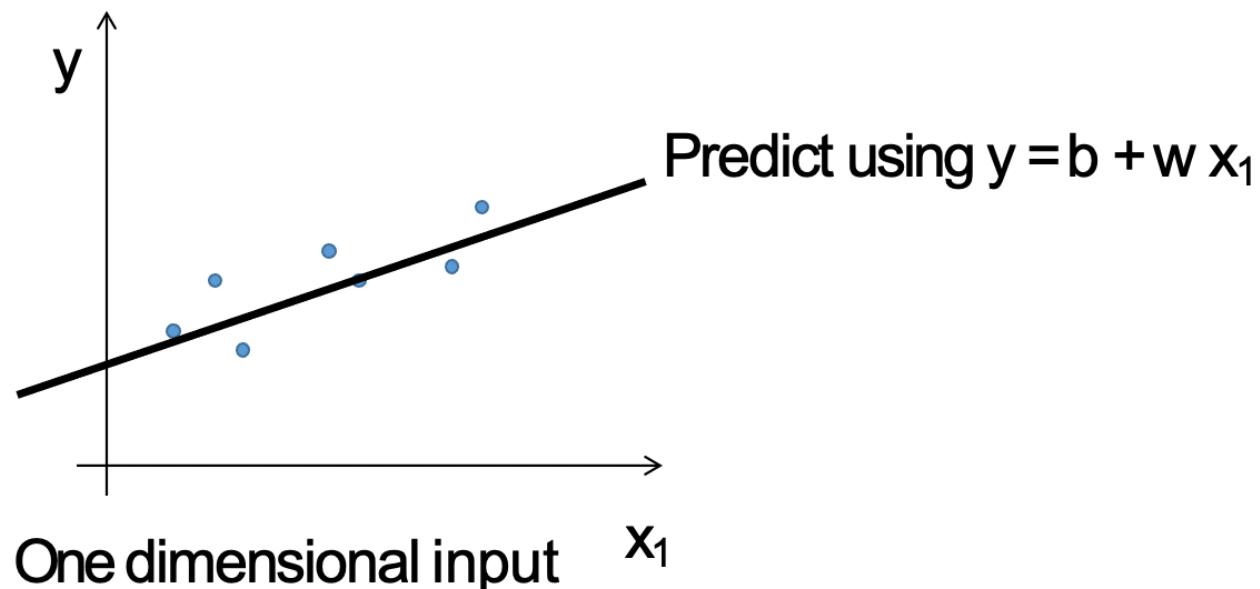
Perceptron effectively minimizing:

Prediction: y^{test}

$$\sum_i \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i))$$

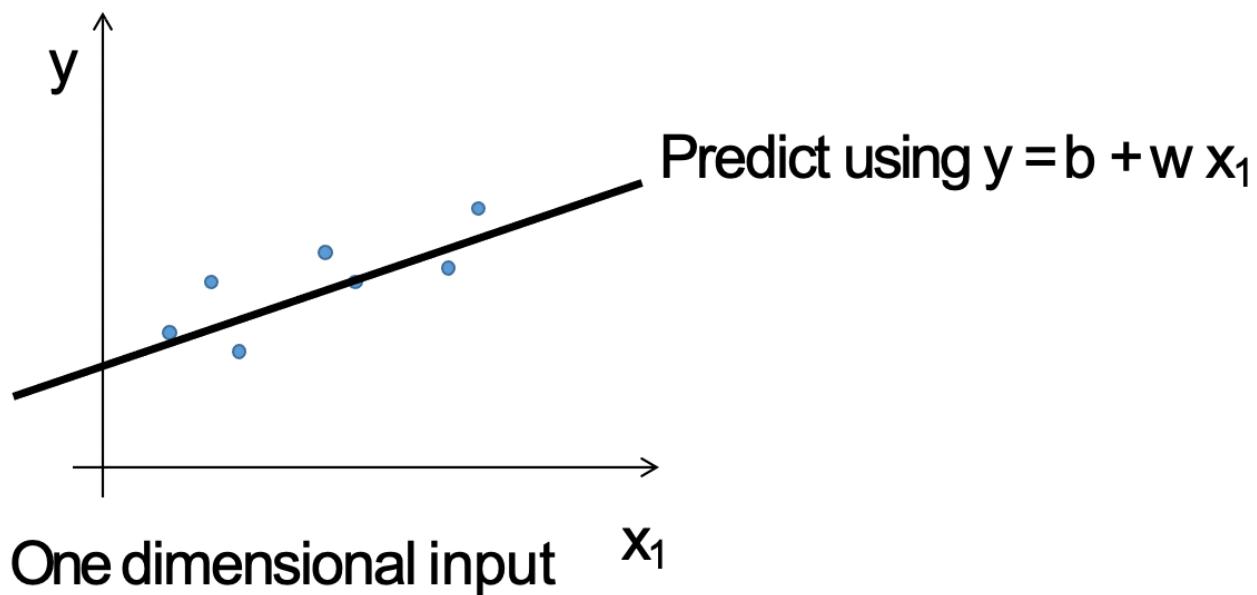
Linear Regression

- ❖ Find a line $w^T x + b$ to approximate real-value output y based on input x
e.g., predict house price next year



Linear Regression

- ❖ Find a line to approximate real-value output
e.g., predict house price next year

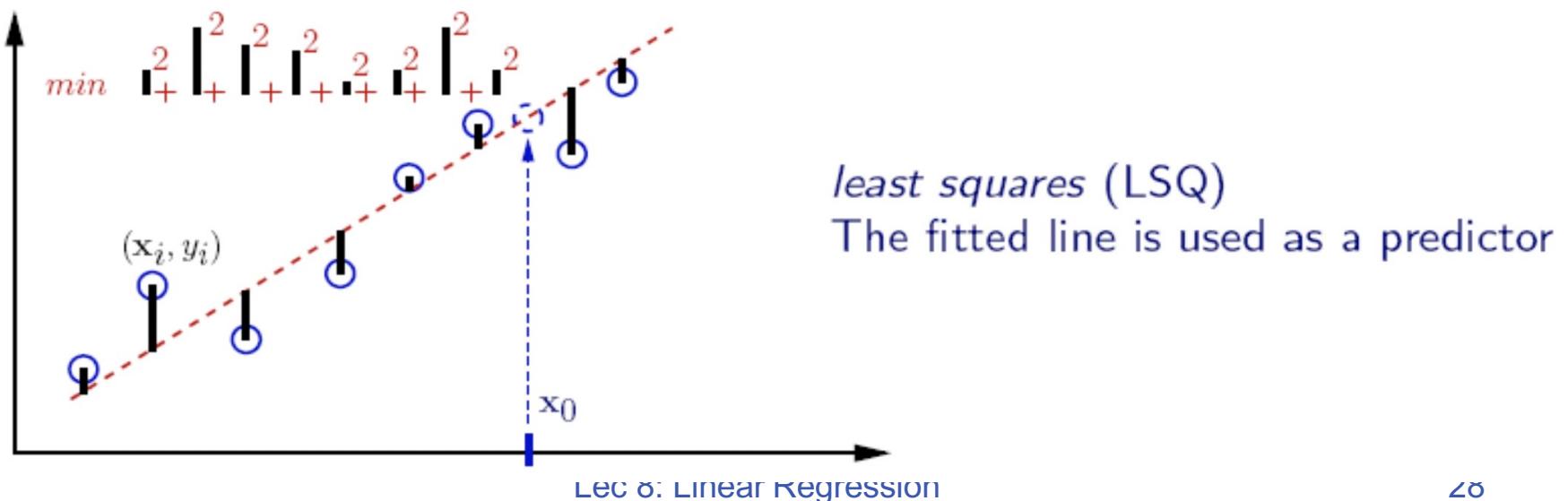


Least Mean Squares (LMS) Regression

Given a dataset $S = \{(x_i, y_i)\}_{i=1..m}, x_i \in \mathbb{R}^n, y \in \mathbb{R}$

$$\arg \min_{w,b} \frac{1}{2} \sum_i^m (y_i - (w^T x_i + b))^2$$

Learning: minimizing mean squared error



Exercise

- ❖ Derive the stochastic gradient descent algorithm for solving LMS regression

$$\arg \min_{w,b} \frac{1}{2} \sum_i^m (y_i - (w^T x_i + b))^2$$

What you will learn today

- ❖ Optimization
 - ❖ Gradient descent
 - ❖ Stochastic gradient descent (SGD)
- ❖ Evaluation Metrics
- ❖ Neural network / Deep learning
 - ❖ Non-linear classifier
 - ❖ Feed-forward neural network
 - ❖ Deep learning architecture

Accuracy and Error Rate

True label	-	-	-	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-
Predicted label	-	-	-	-	+	-	+	-	-	-	-	-	+	-	+	-	-	-

Error rate = 3/16

Accuracy = 13/16

Accuracy and Error Rate

True label	-	-	-	-	+	-	-	-	-	-	-	+	-	-	-	+	-	-	-
Predicted label	-	-	-	-	+	-	+	-	-	-	-	-	+	-	-	+	-	-	-

$$\text{Error rate} = 3/16 = 19\%$$

$$\text{Accuracy} = 13/16 = 81\%$$

When data is unbalanced, check the performance of majority baseline

True label	-	-	-	-	+	-	-	-	-	-	-	+	-	-	-	+	-	-	-
Predicted label	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Confusion Matrix

True label	-	-	-	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-
Predicted label	-	-	-	-	+	-	+	-	-	-	-	-	+	-	+	-	-	-

		True Label Positive	True Label Negative
Predicted Label Positive	2 True Positive (TP)	2 False Positive (FP)	
	1 False Negative (FN)	16 True Negative (TN)	

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FN} + \text{FP})$$

Precision, Recall

True label	-	-	-	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-
Predicted label	-	-	-	-	+	-	+	-	-	-	-	-	+	-	+	-	-	-

		True Label Positive	True Label Negative
Predicted Label Positive	2 True Positive (TP)	2 False Positive (FP)	
Predicted Label Negative	1 False Negative (FN)	16 True Negative (TN)	

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FN} + \text{FP})$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

F1 Score

- ❖ Harmonic mean of precision and recall:

$$\frac{1}{F_1} = \left(\frac{1}{P} + \frac{1}{R} \right) / 2$$

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

What you will learn today

- ❖ Optimization
 - ❖ Gradient descent
 - ❖ Stochastic gradient descent (SGD)
- ❖ Evaluation Metrics
- ❖ Neural network / Deep learning
 - ❖ Non-linear classifier
 - ❖ Feed-forward neural network
 - ❖ Deep learning architecture

Checkpoint: The bigger picture

- ❖ Supervised learning: instances, concepts, and hypotheses

- ❖ Specific learners

- ❖ Decision trees

- ❖ K-NN

- ❖ Perceptron

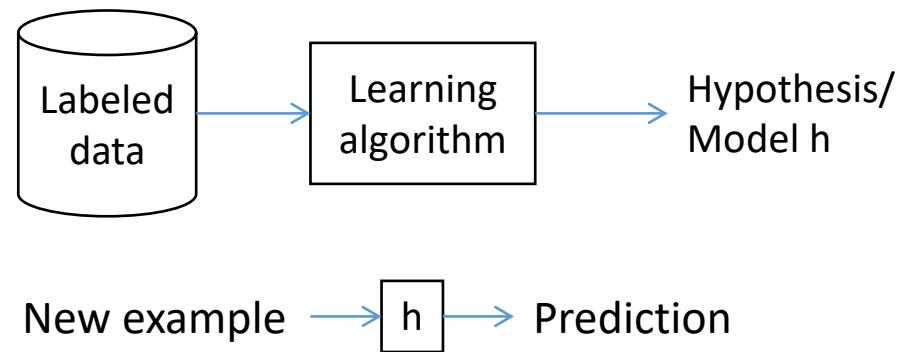
- ❖ Logistic regression

- ❖ General ML ideas

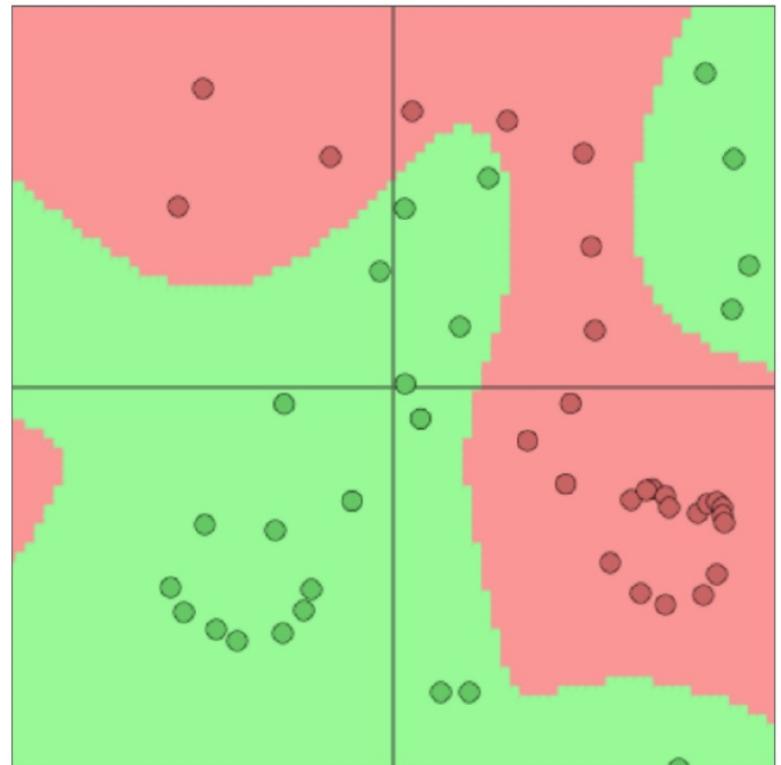
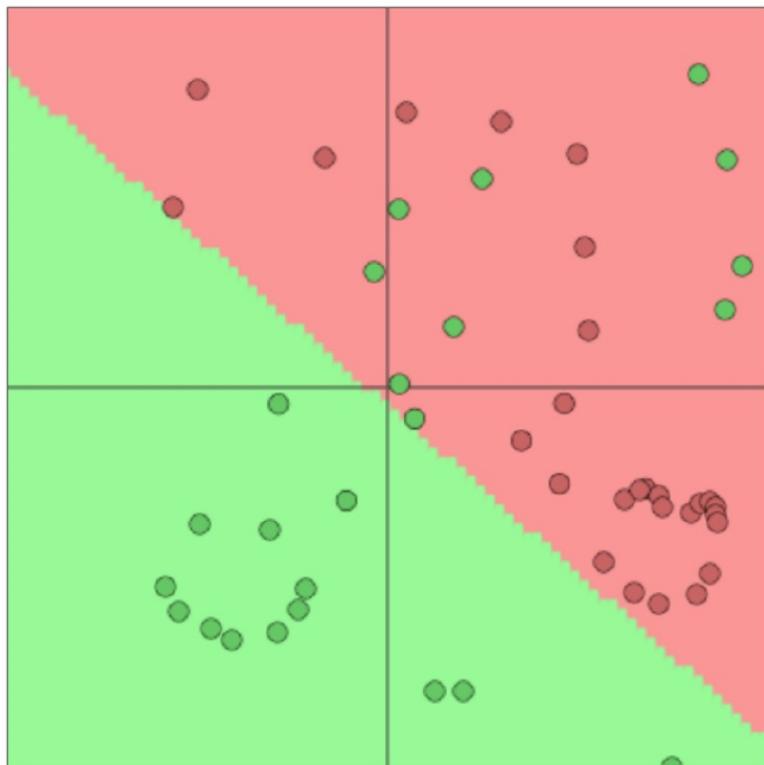
- ❖ Feature vectors

- ❖ Overfitting

- ❖ Probabilistic model

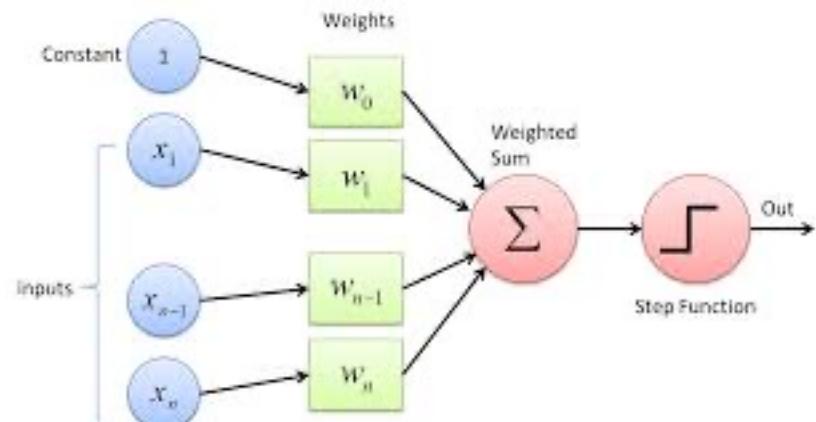
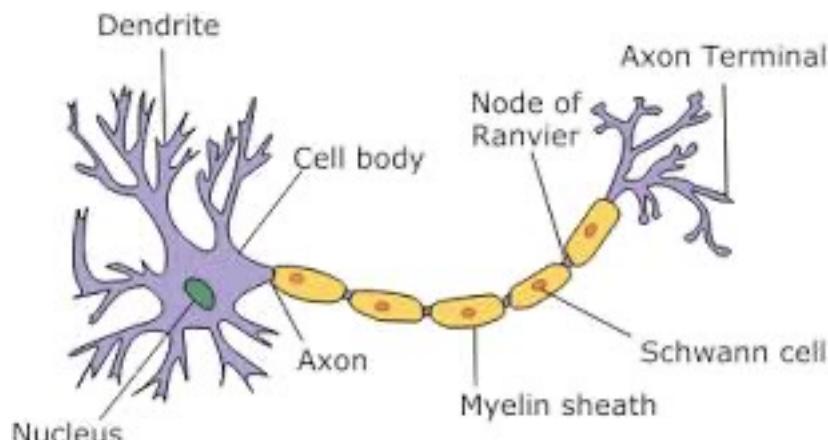


Non-Linear Decision Boundary

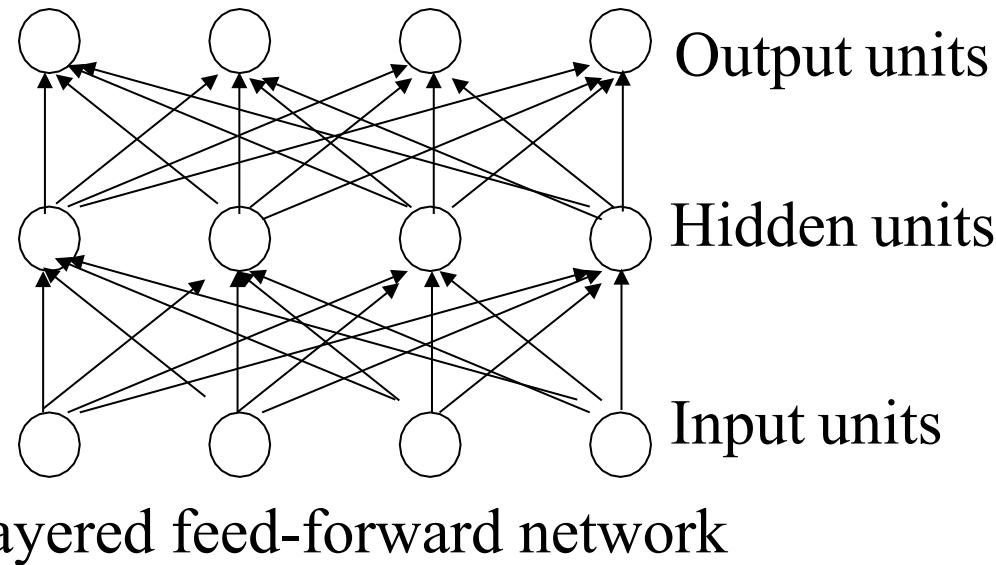


Neural Networks

- Design to mimic the brain.
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure



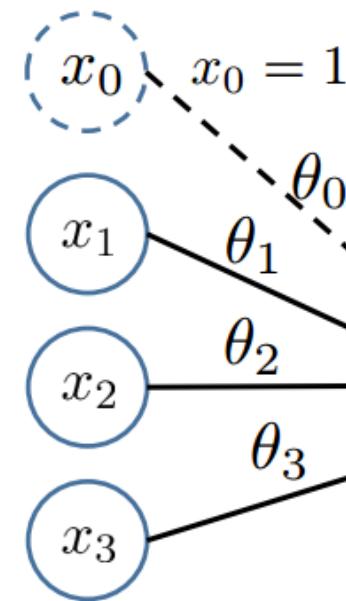
Feed-forward neural network



- Neural networks are made up of **nodes** or **units**, connected by **links**
- Each link has an associated **weight** and **activation level**
- Each node has an **input function** (typically summing over weighted inputs), an **activation function**, and an **output**

Neuron Model Example: Logistic Unit

“bias unit”

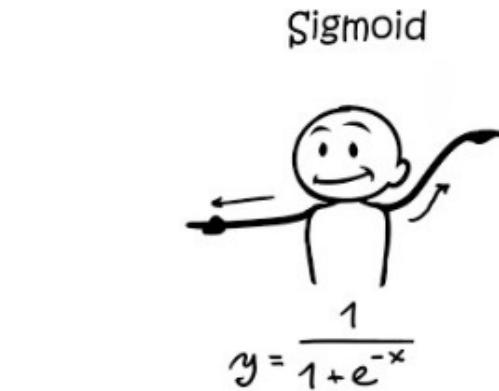
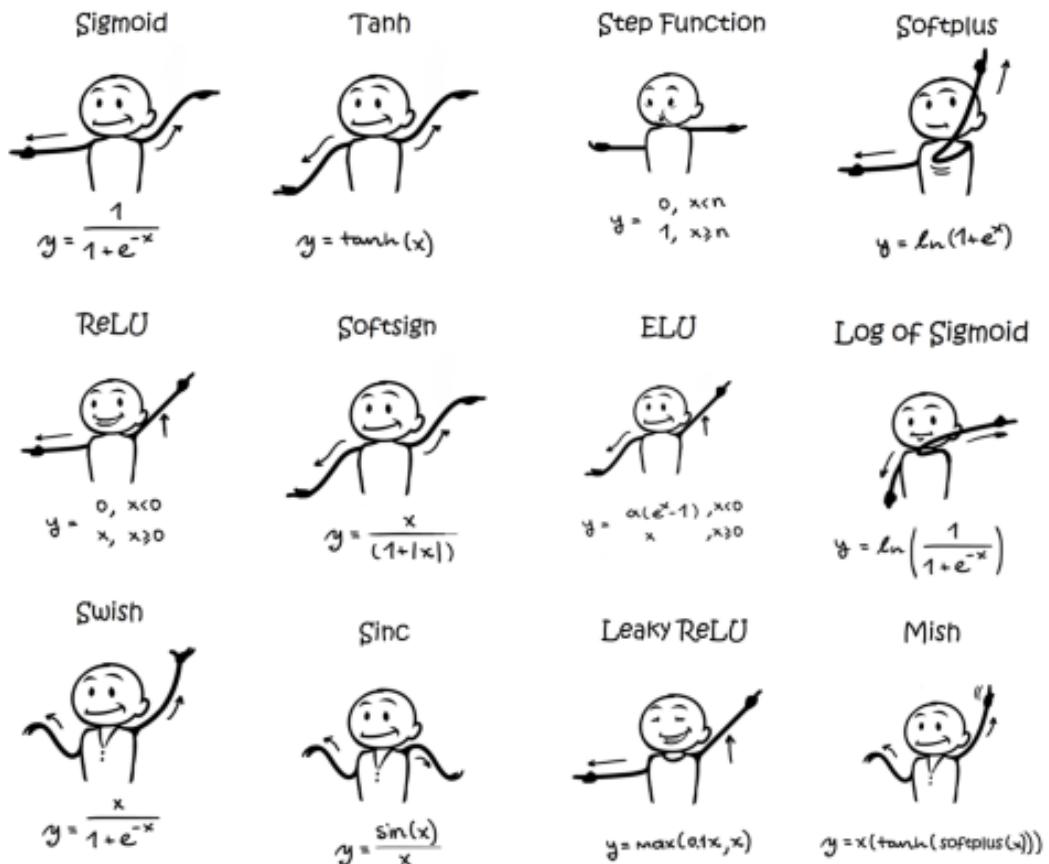


$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\theta}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$

Activation function

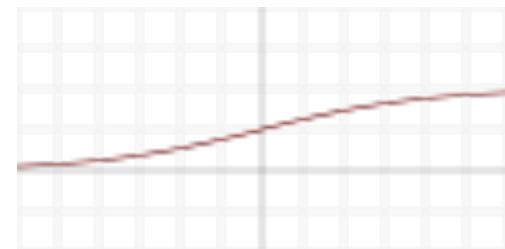


<https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions/>
by Sefik Ilkin Serengil

Activation functions

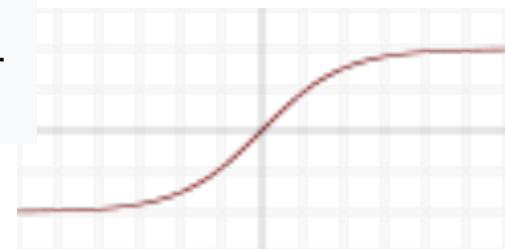
- ❖ sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}$$



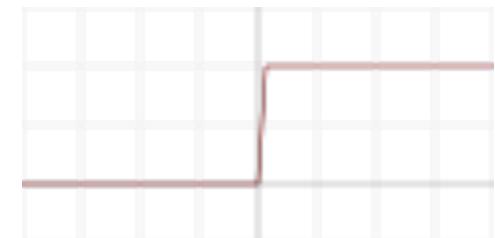
- ❖ hyperbolic tangent

$$\tanh(x) \doteq \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



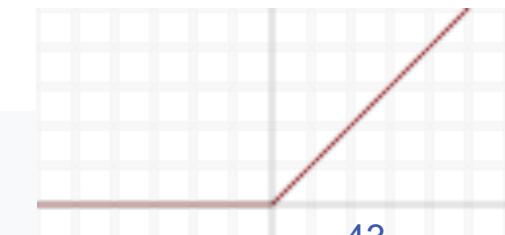
- ❖ step function

$$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

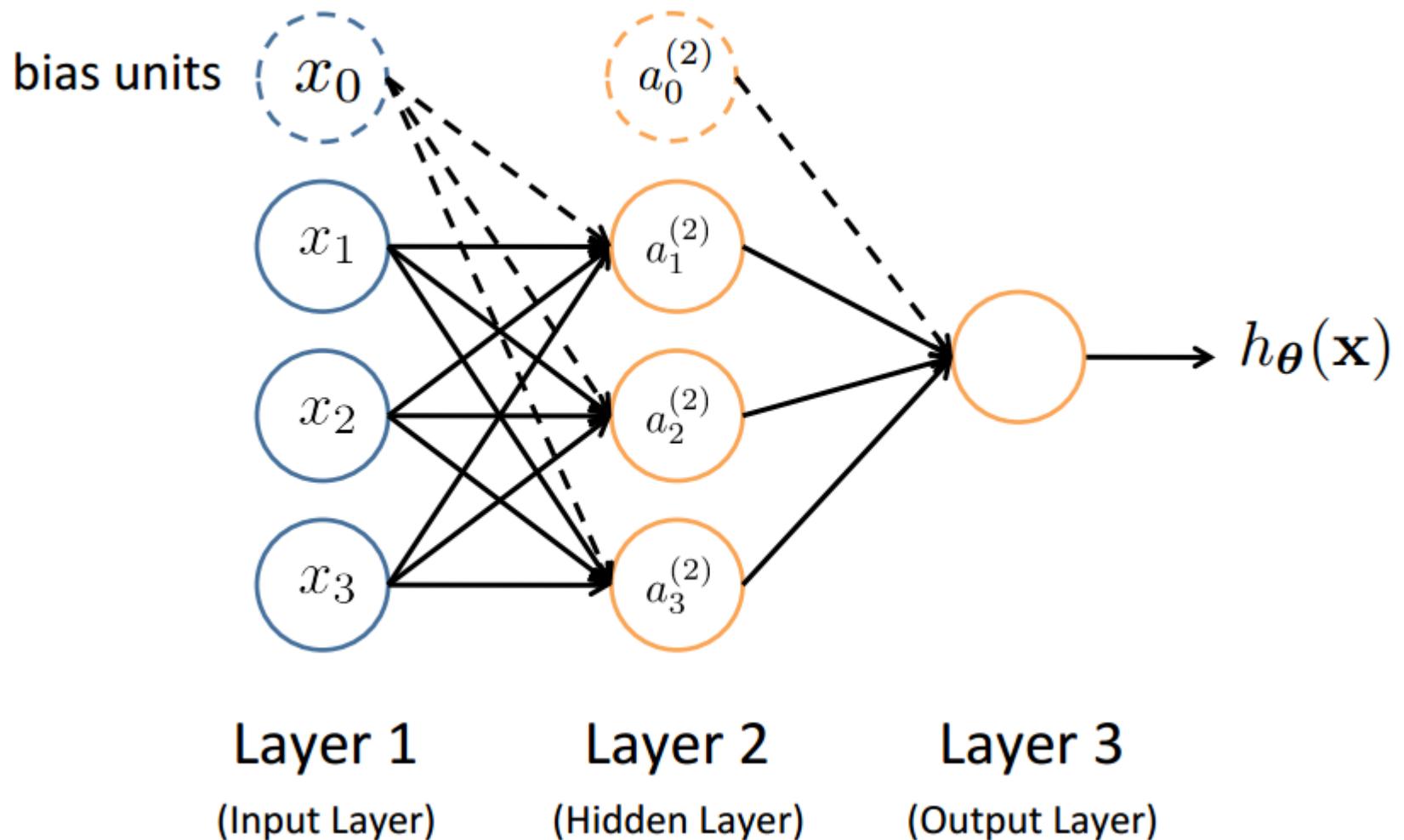


- ❖ Rectified linear unit (ReLU)

$$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

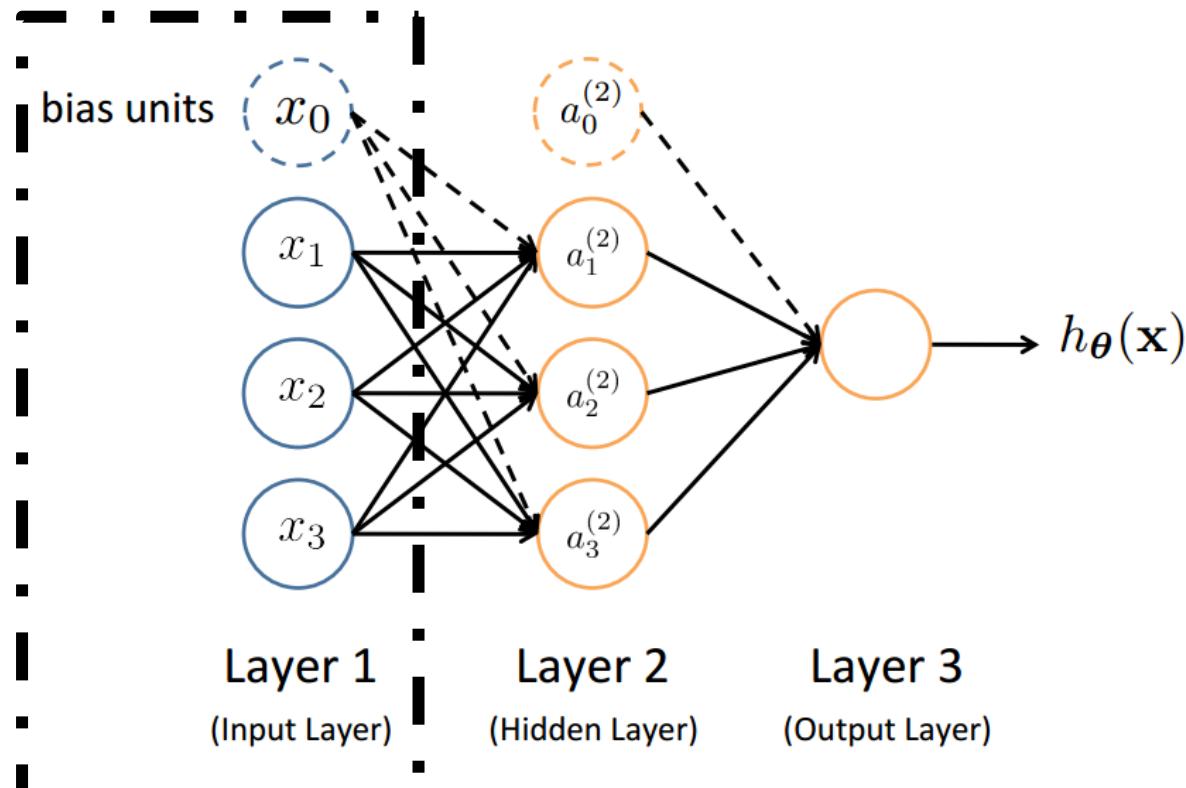


Neural Network



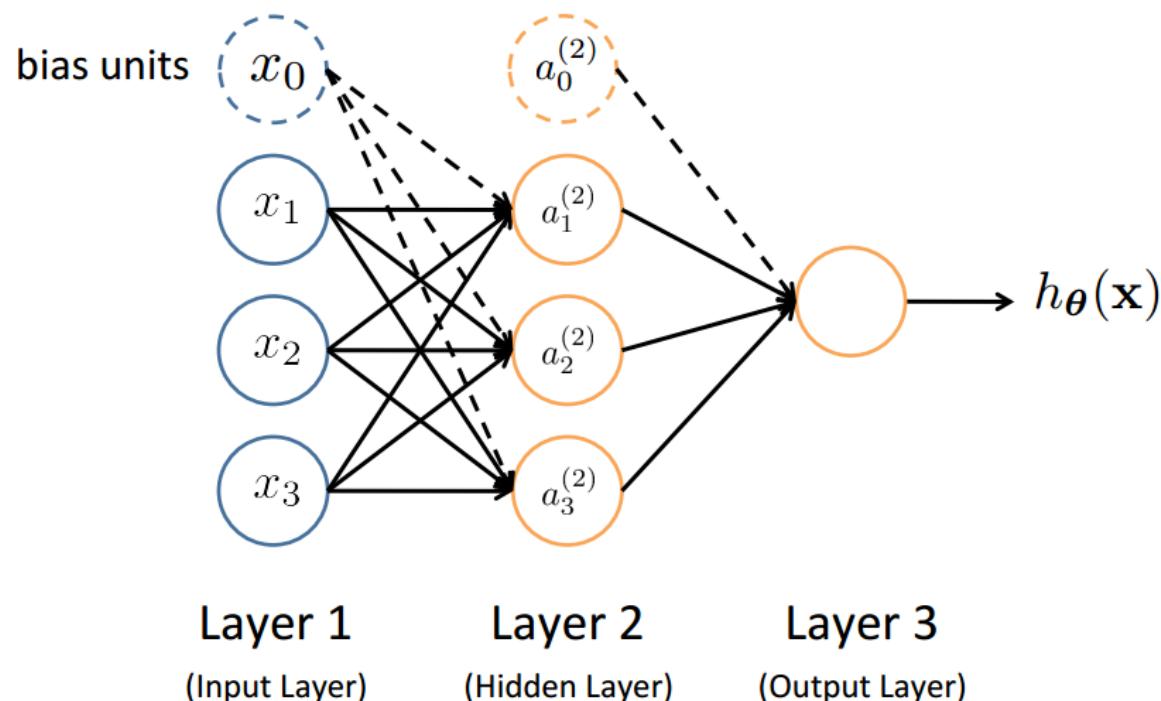
Feed-Forward Process

- Input layer units are set by some exterior function (think of these as **sensors**), which causes their output links to be **activated** at the specified level

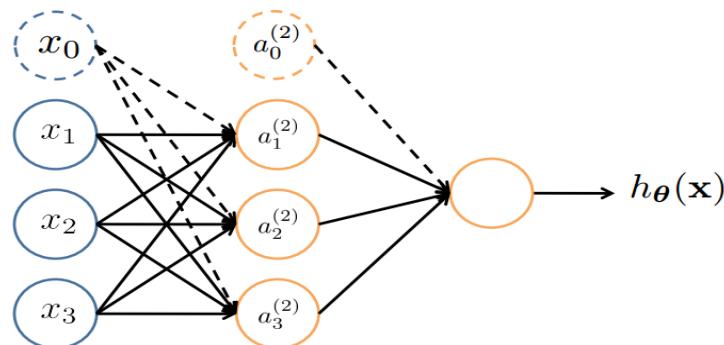


Feed-Forward Process

- Working forward through the network, the **input function** of each unit is applied to compute the input value
- The **activation function** transforms this input function into a final value



Neural Network



$a_i^{(j)}$ = “activation” of unit i in layer j
 $\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$,
then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

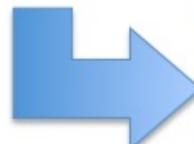
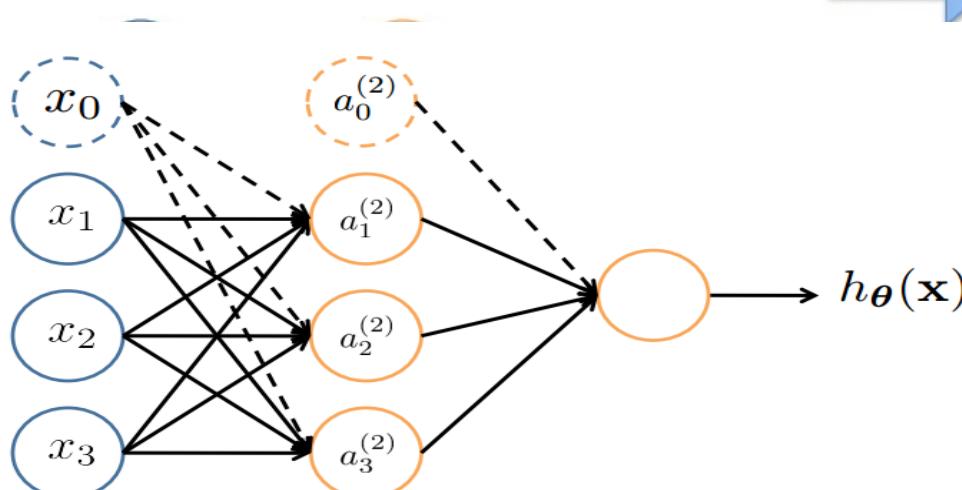
Vectorization

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left(z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left(z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left(z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left(z_1^{(3)} \right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Exercise

- ❖ Why do we need non-linear activation functions?
- ❖ What happen if $g(z) = z$

Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

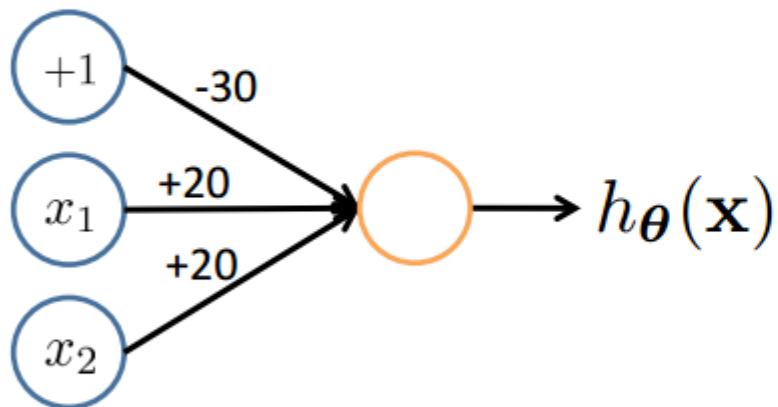
$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Non-Linear Representations

Simple example: AND

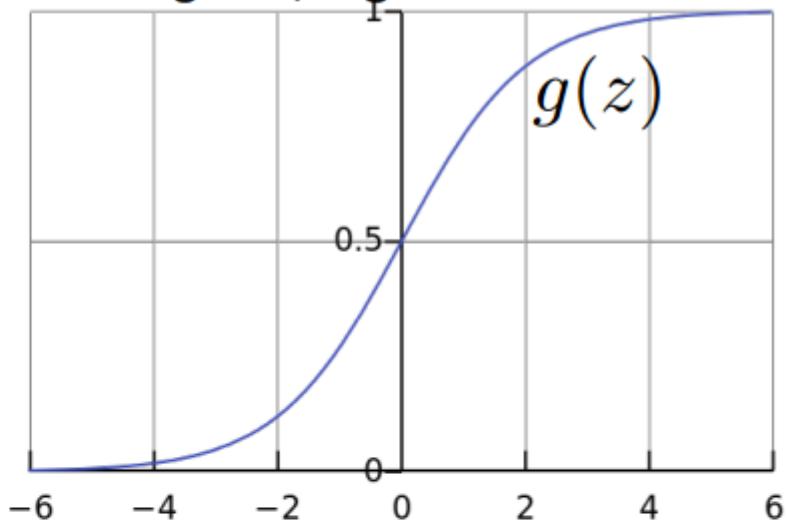
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$

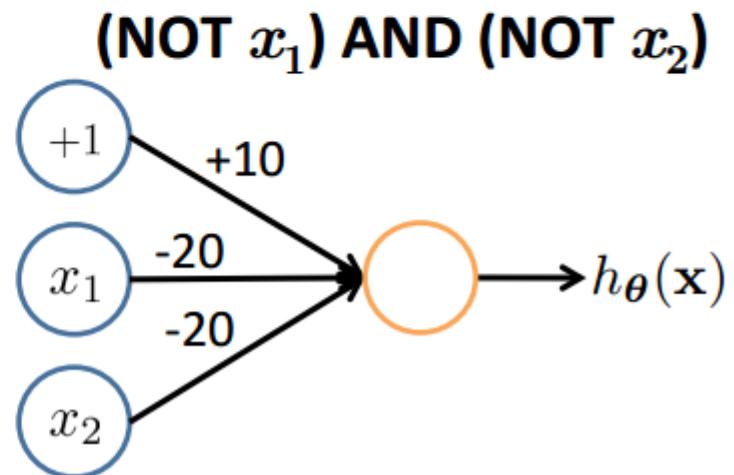
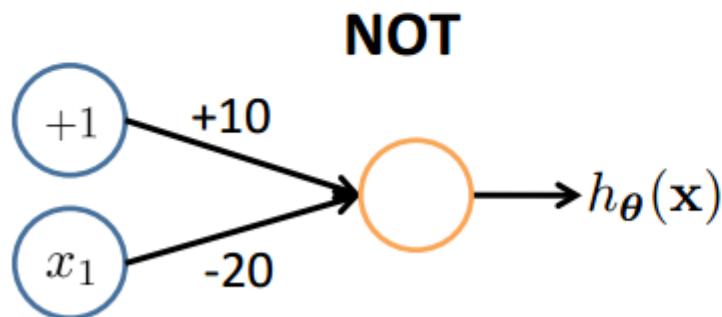
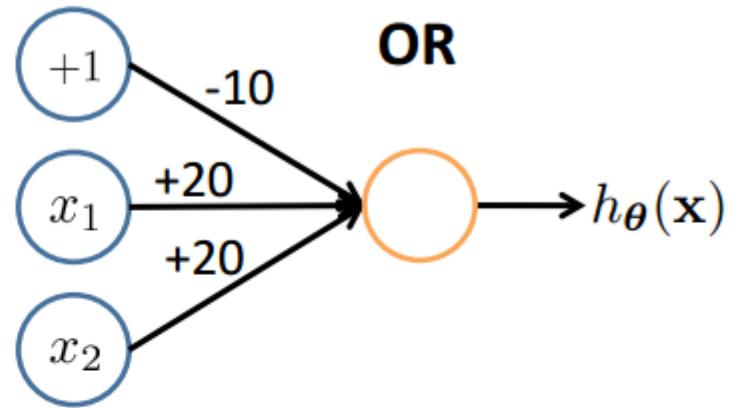
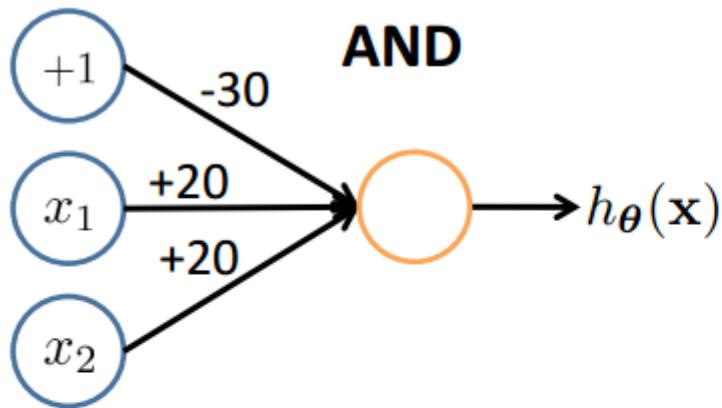


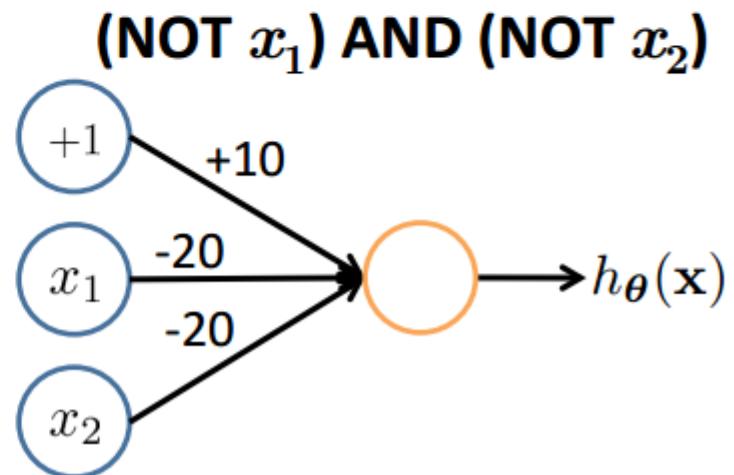
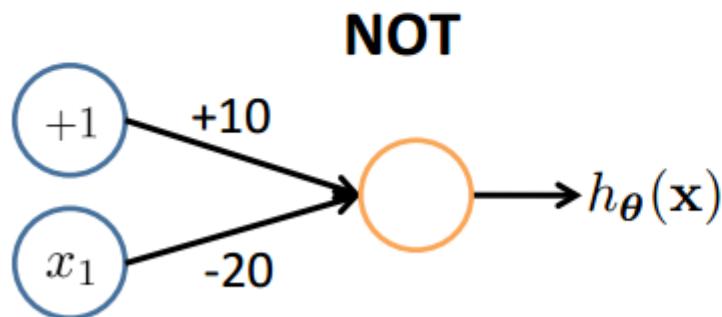
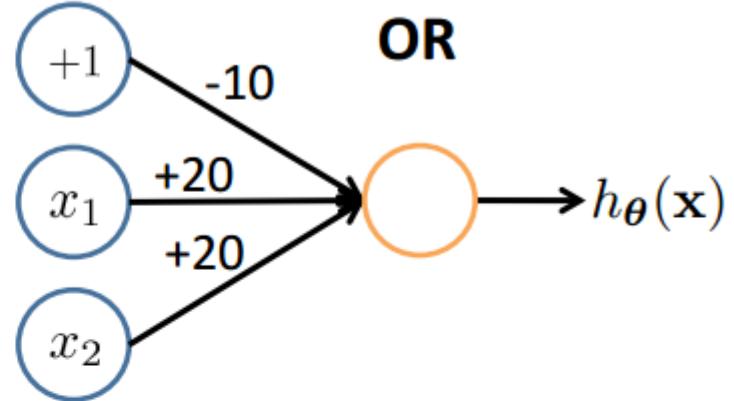
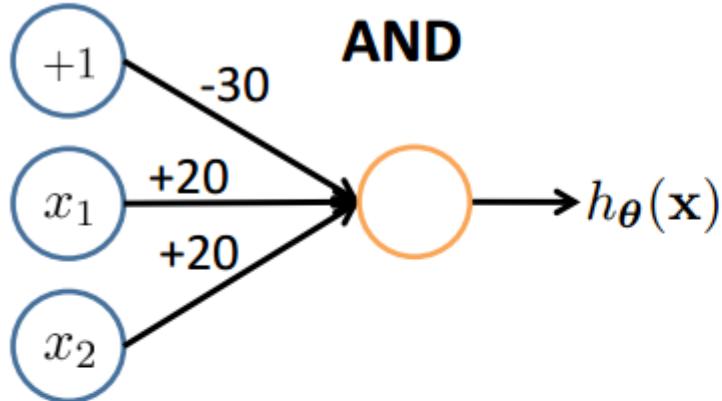
$$h_{\theta}(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

Logistic / Sigmoid Function

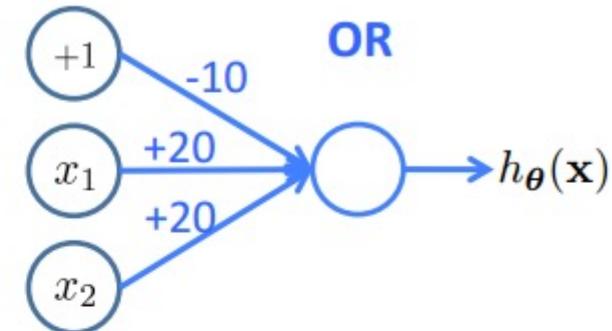
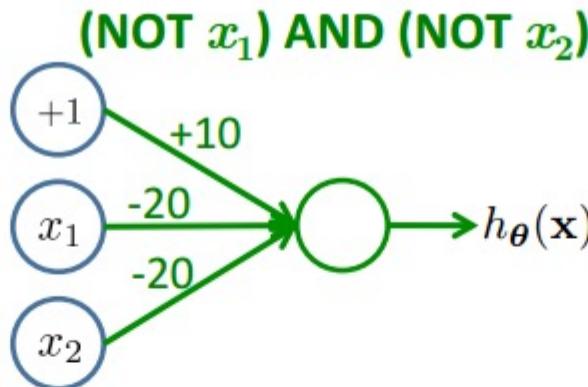
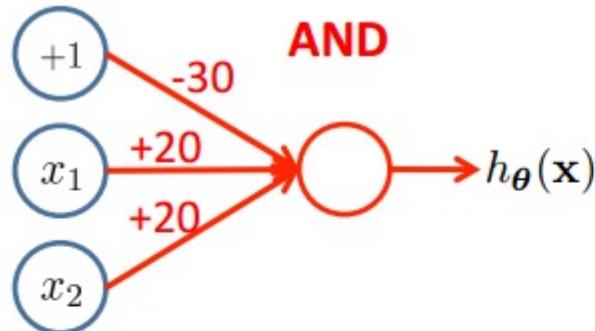


x_1	x_2	$h_{\theta}(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

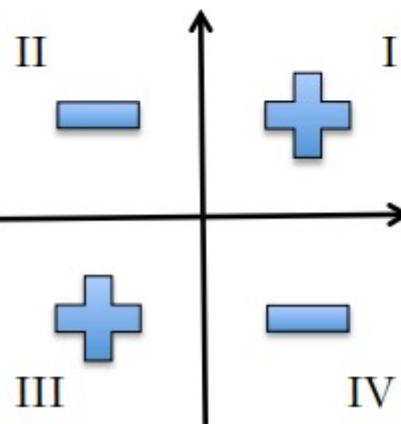




Combining Representations to Create Non-Linear Functions

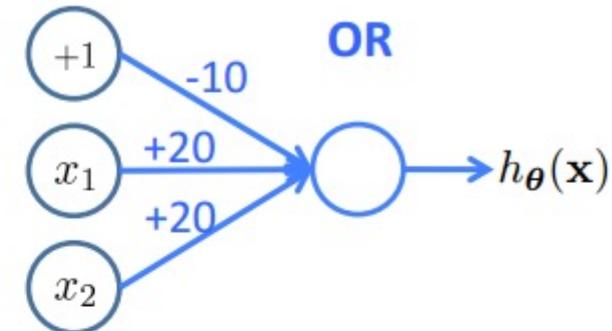
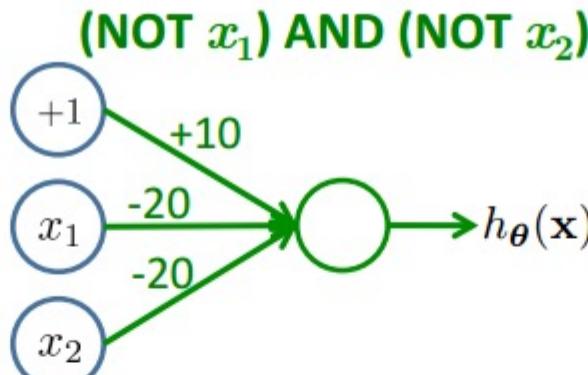
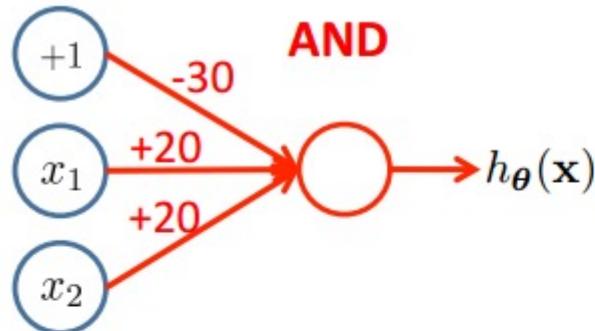


XNOR

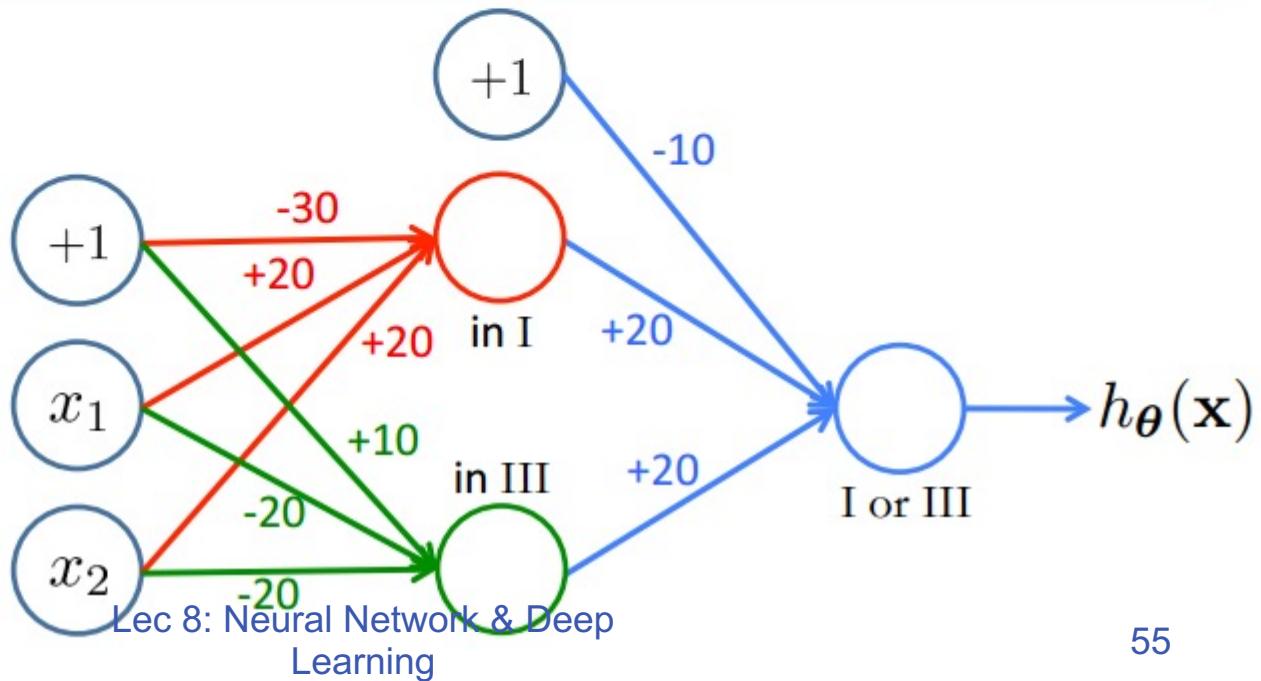
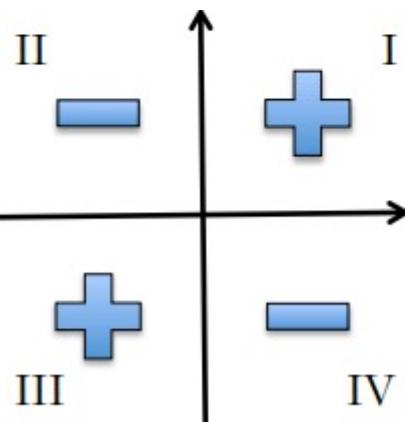


XNOR

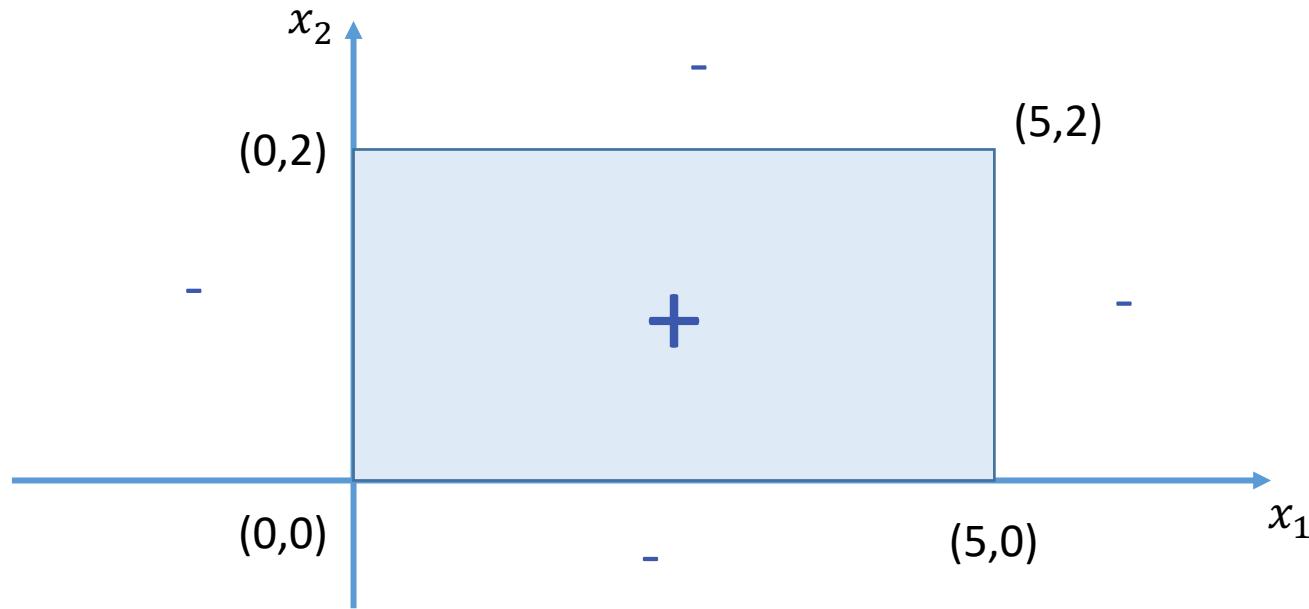
Combining Representations to Create Non-Linear Functions



XNOR



Exercise



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

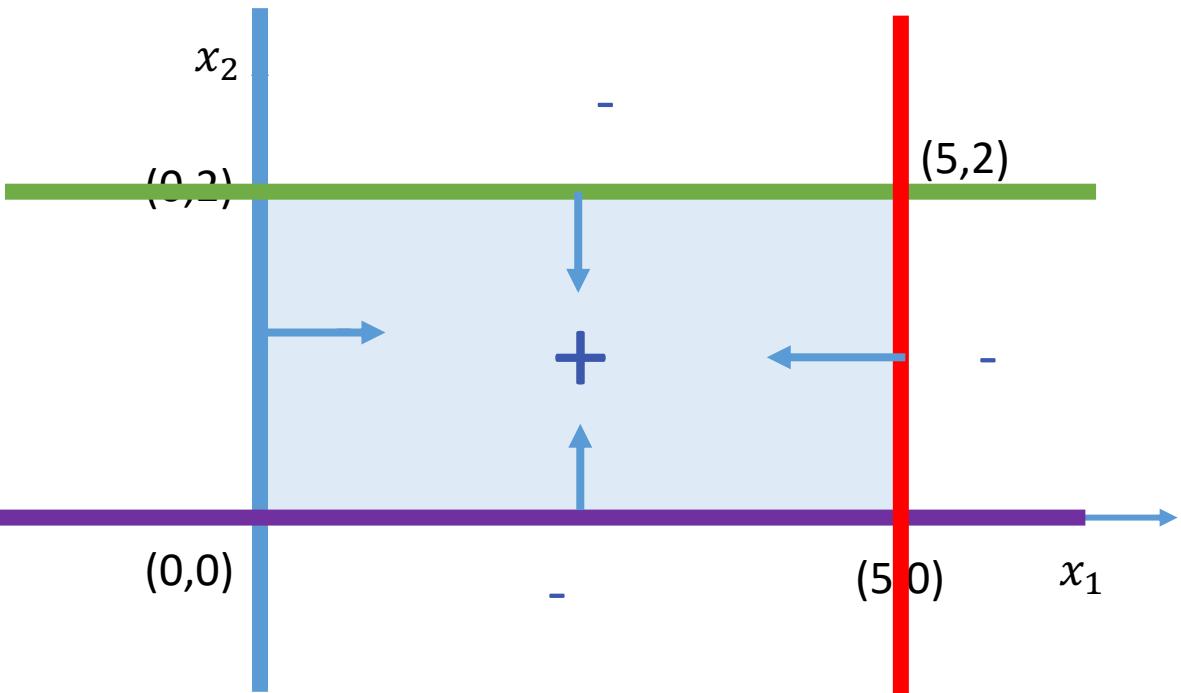
$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

If all the samples inside the rectangle are positive;
otherwise are negative

Show a feedforward NN can classify all the samples correctly
For simplicity, we assume $g(z)$ is a step function.
What are $\Theta^{(1)}$ and $\Theta^{(2)}$

Exercise



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

If all the samples inside the rectangle are positive;
otherwise are negative

Show a feedforward NN can classify all the samples correctly
For simplicity, we assume $g(z)$ is a step function.
What are $\Theta^{(1)}$ and $\Theta^{(2)}$

$$\begin{cases} x_1 & > 0 \\ 5 - x_1 & > 0 \\ x_2 & > 0 \\ -x_2 & > 0 \end{cases}$$

$$\Rightarrow \Theta^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ 5 & -1 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & -1 \end{bmatrix}$$

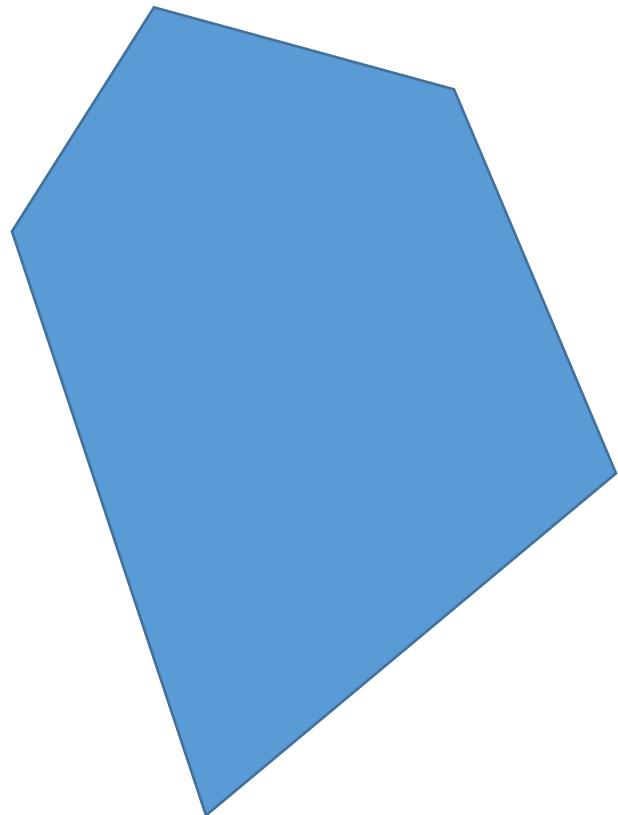
$$\Theta_0^{(1)} \Theta_1^{(1)} \Theta_2^{(1)}$$

$$-3.5 + a_1 + a_2 + a_3 + a_4 > 0$$

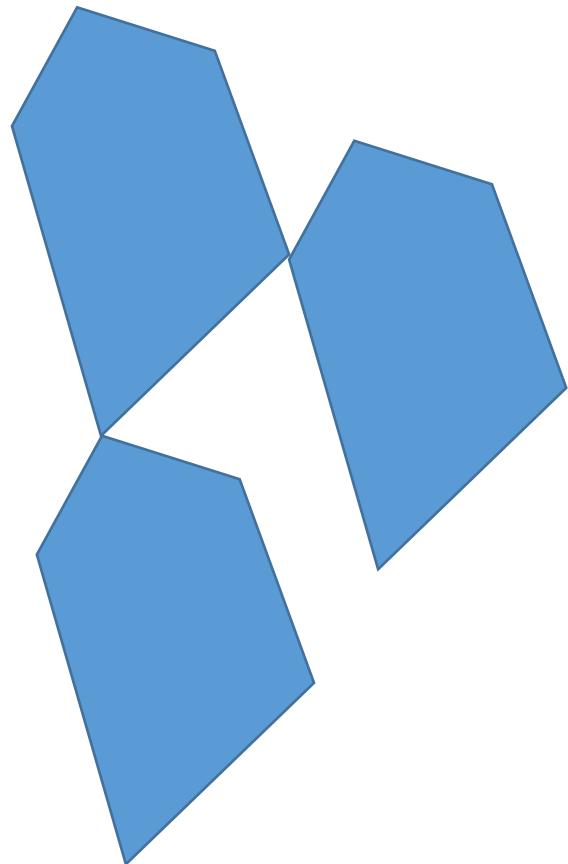
$$\Theta^{(2)} = [-3.5 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$\Theta_0^{(2)} \Theta_1^{(2)} \Theta_2^{(2)} \Theta_3^{(2)} \Theta_4^{(2)}$$

Arbitrary Decision Boundary

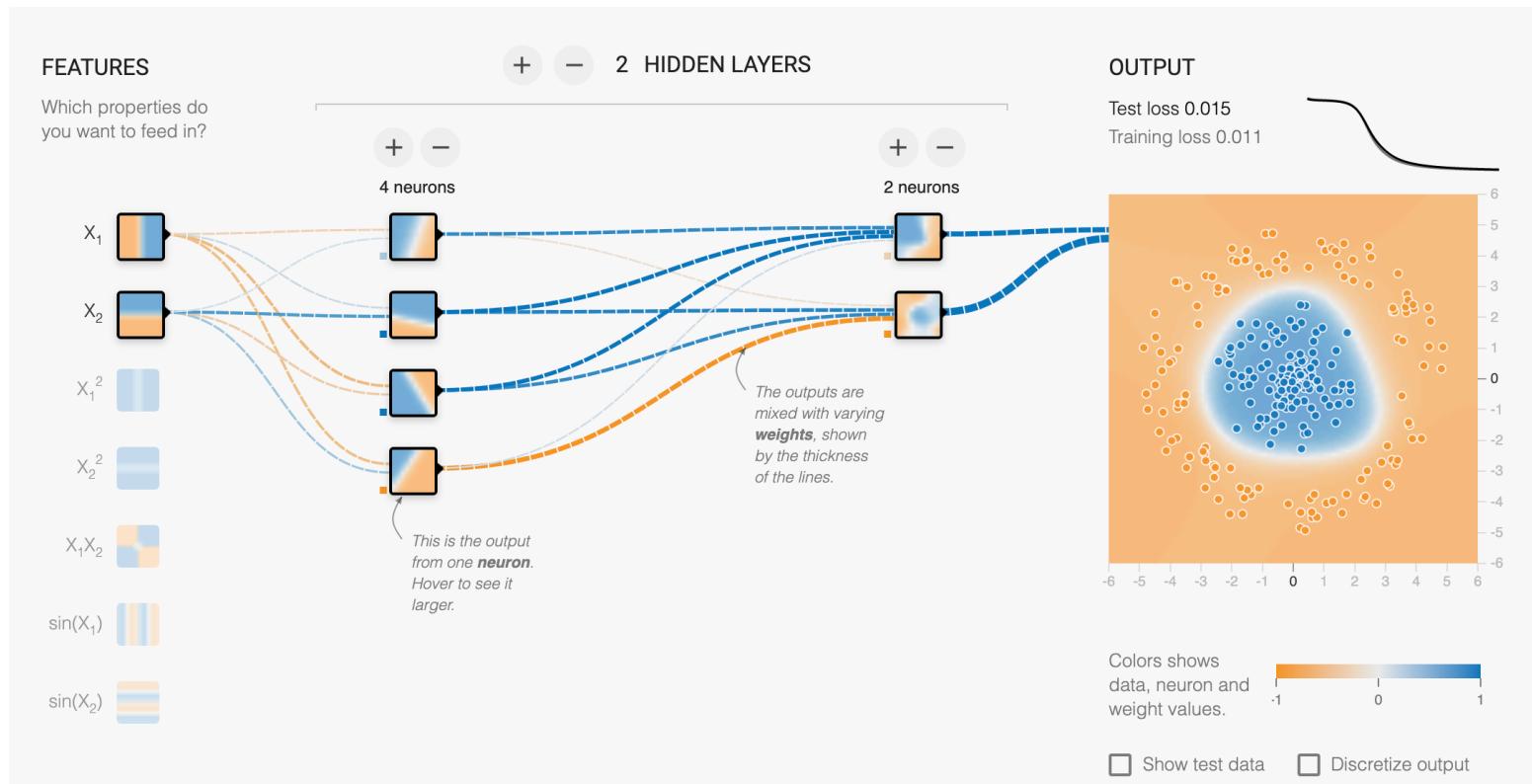


Arbitrary Decision Boundary



Neural Network Training Animation

❖ <https://playground.tensorflow.org/>



Lecture 9: Deep Learning Learning Theory

Fall 2022

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcements

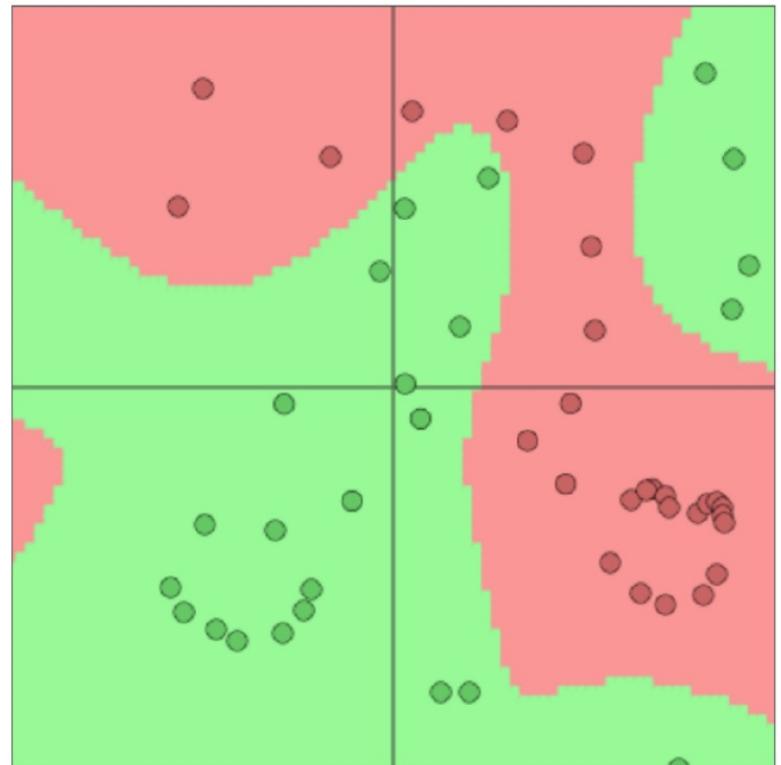
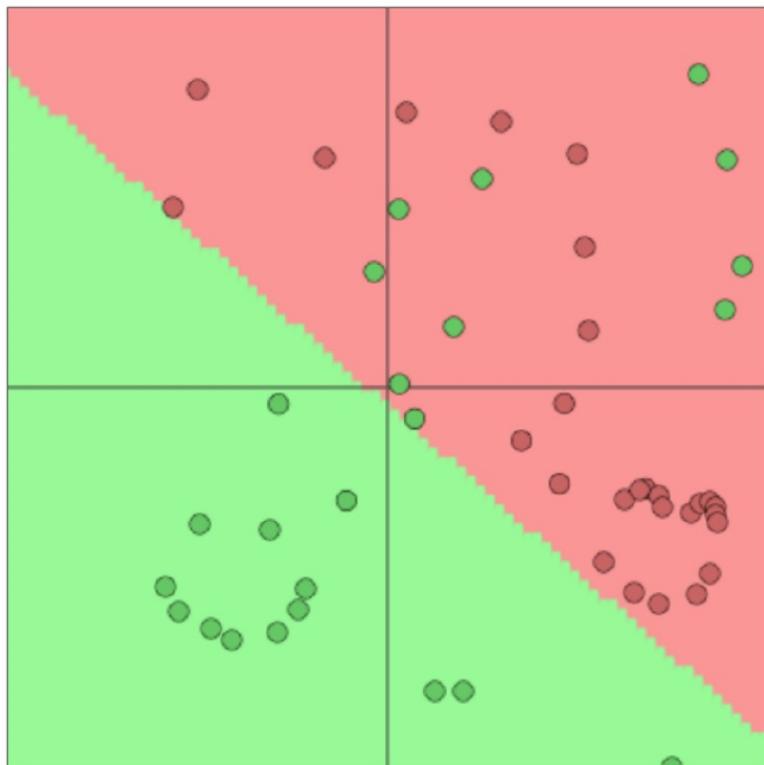
- ❖ Midterm postpones to 11/3
 - ❖ The practice exam will be posted
- ❖ Hw1 is due next Tue!
- ❖ Hw 2 & Quiz 3 will be released tomorrow



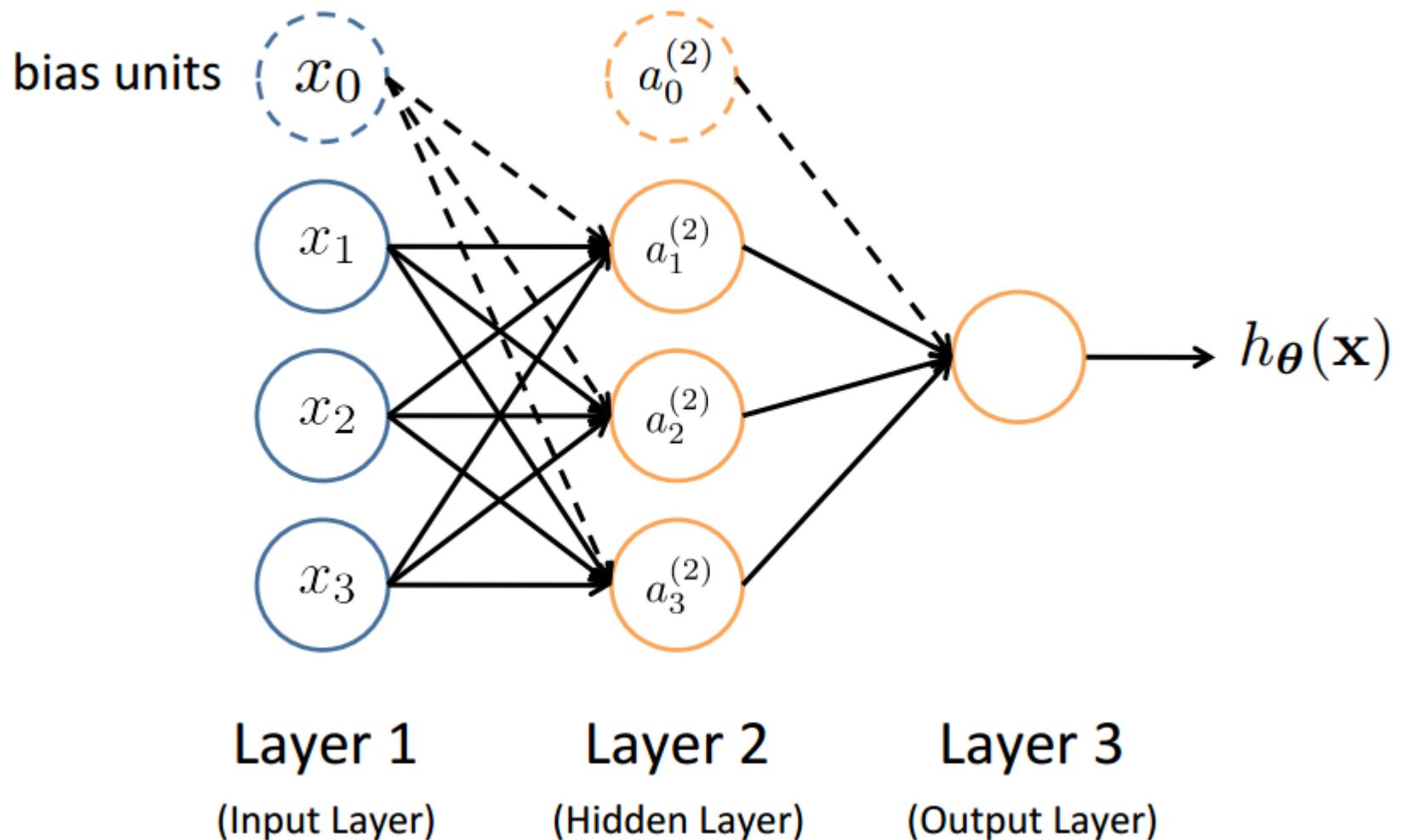
What you will learn today

- ❖ Neural network / Deep learning
 - ❖ Non-linear classifier
 - ❖ Feed-forward neural network
 - ❖ Back Propagation
 - ❖ Deep learning architecture

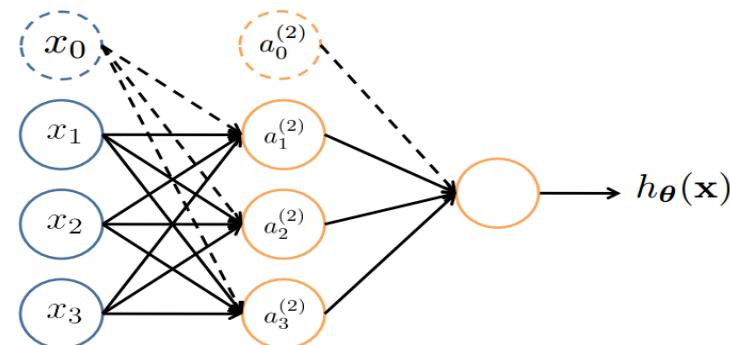
Non-Linear Decision Boundary



Neural Network



Neural Network



$a_i^{(j)}$ = “activation” of unit i in layer j
 $\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$,
then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

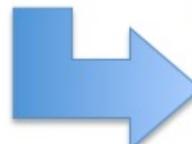
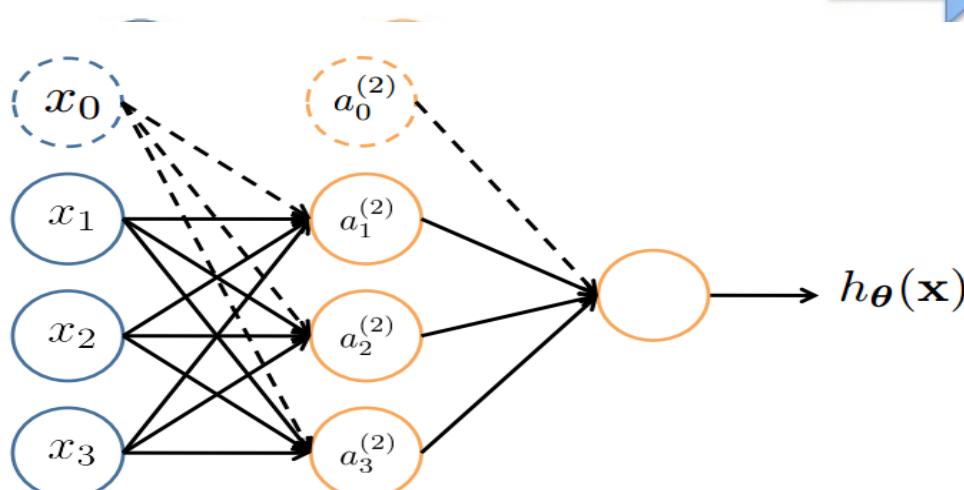
Vectorization

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left(z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left(z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left(z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left(z_1^{(3)} \right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

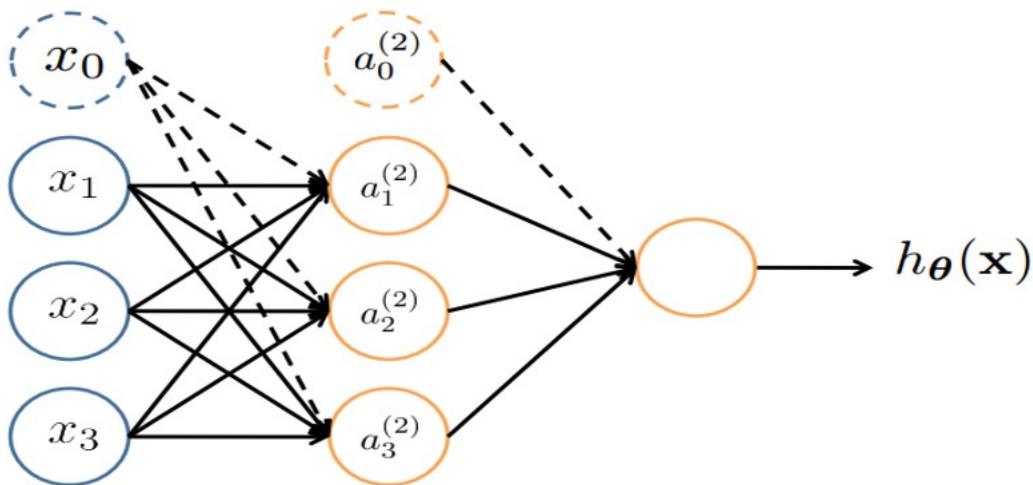
$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Example



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

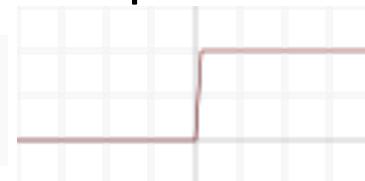
$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Let $\Theta^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{bmatrix}$

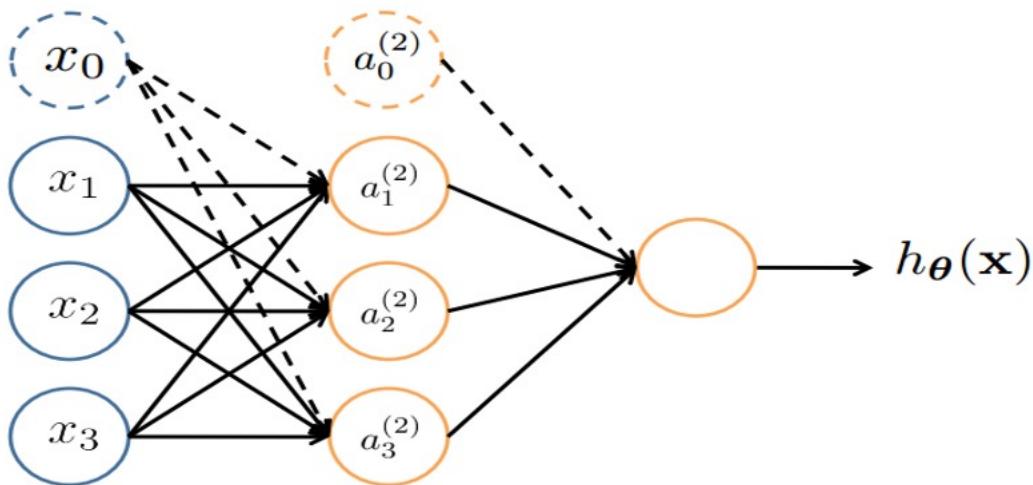
$\Theta^{(2)} = [0 \quad 1 \quad 1 \quad 0]$
 $g(z)$ is a step function

$$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$



What is the output of $x = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix}$?

Example



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

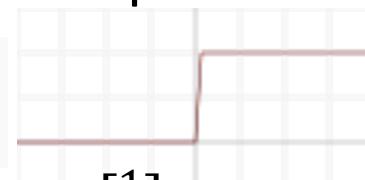
$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

$$\text{Let } \Theta^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{bmatrix}$$

$$\Theta^{(2)} = [0 \quad 1 \quad 1 \quad 0]$$

$g(z)$ is a step function

$$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$



What is the output of $x = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix}$?

$$z^{(2)} = \begin{bmatrix} 2 \\ 2 \\ 5 \end{bmatrix} \quad a^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad z^{(3)} = 2 \quad h_{\Theta}(x) = 1$$

Exercise

- ❖ Why do we need non-linear activation functions?
- ❖ What happen if $g(z) = z$

Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

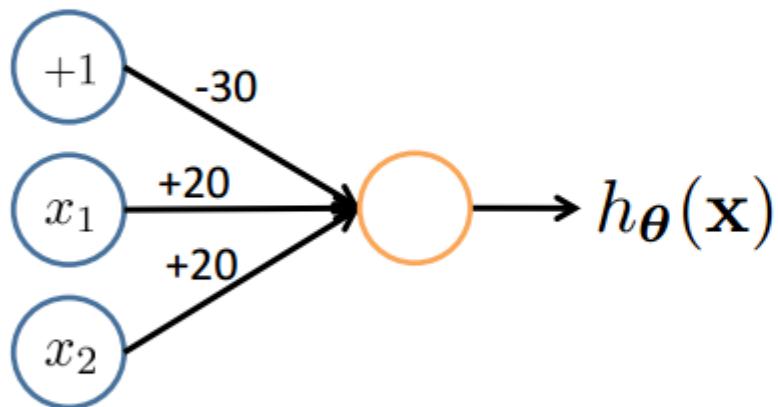
$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Non-Linear Representations

Simple example: AND

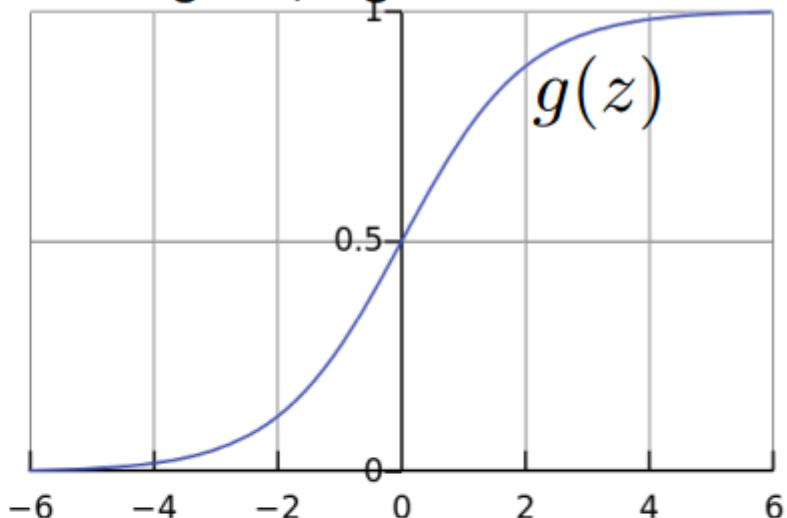
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$

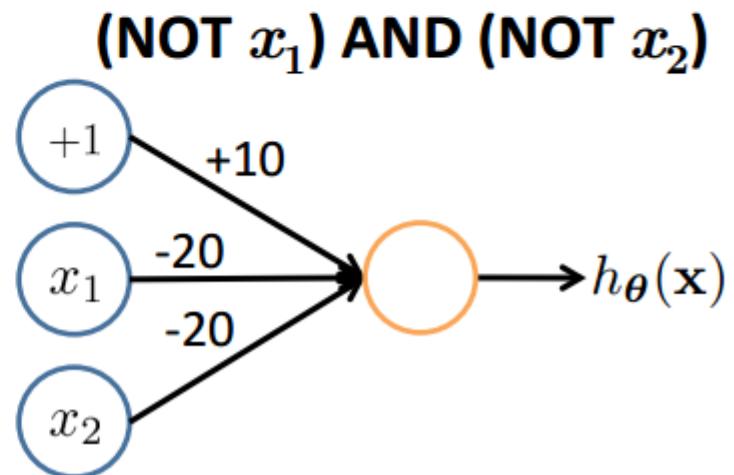
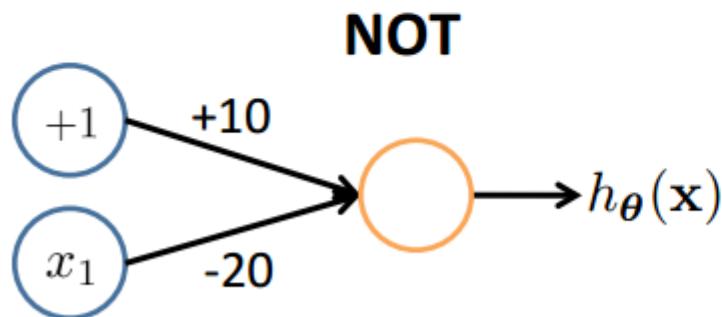
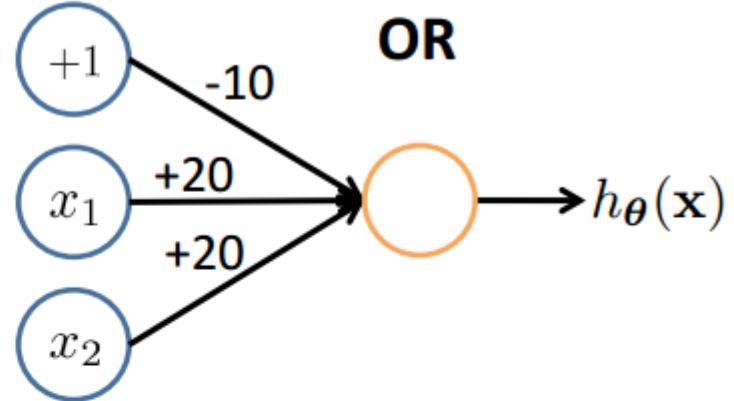
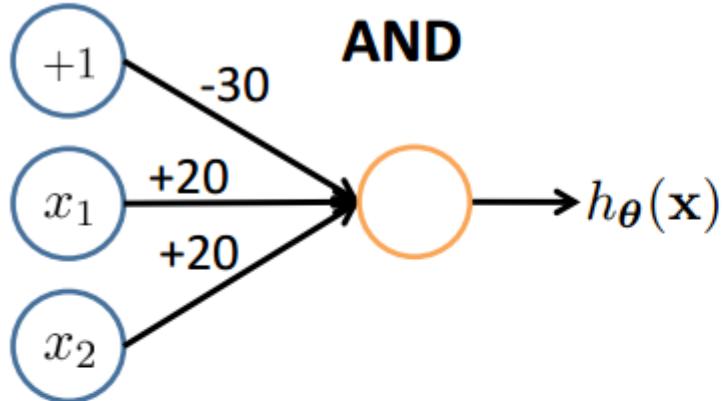


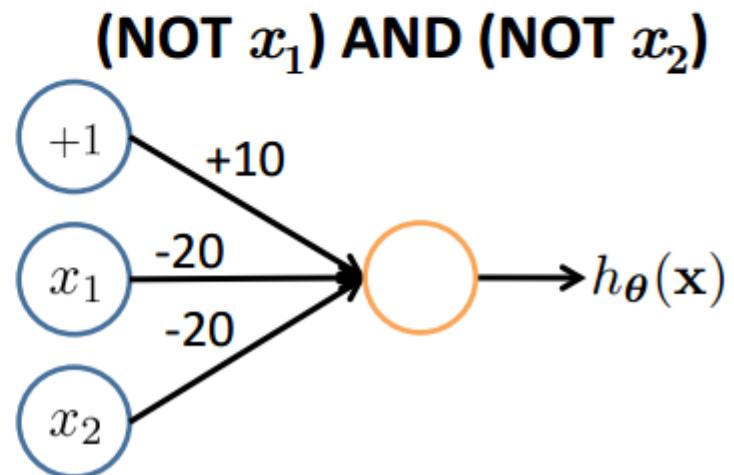
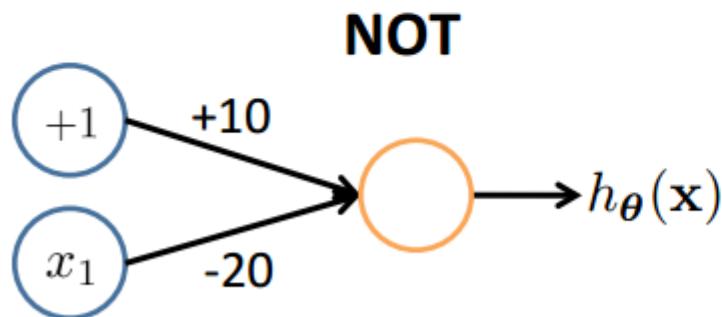
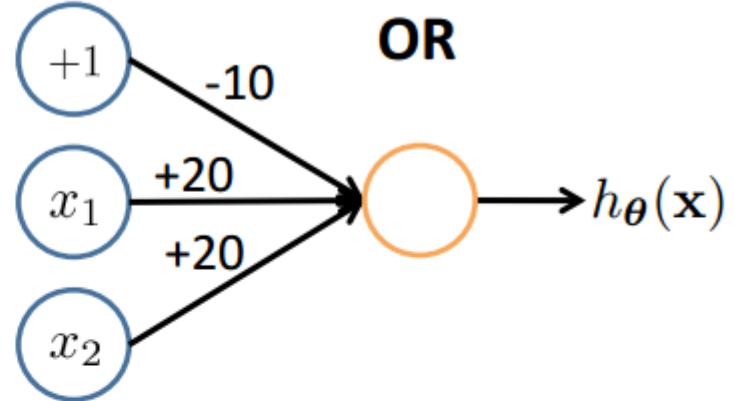
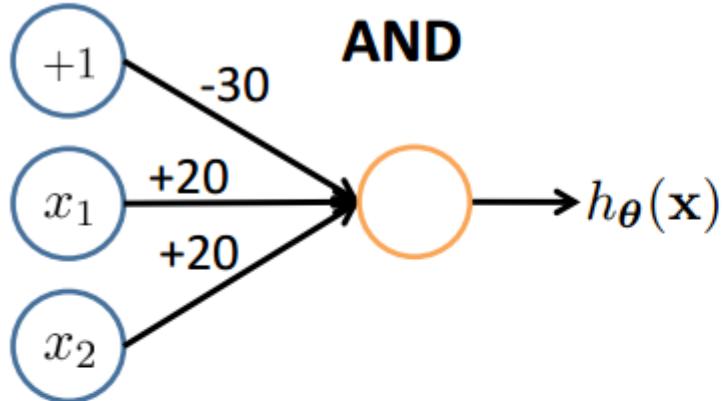
$$h_{\theta}(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

Logistic / Sigmoid Function

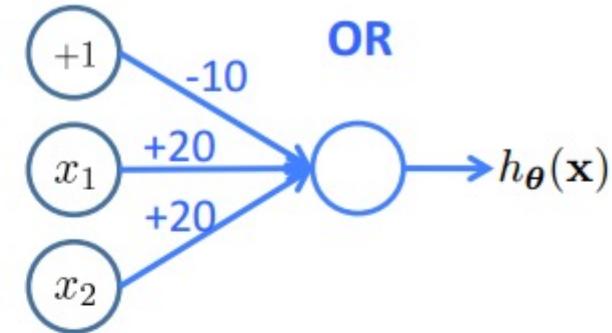
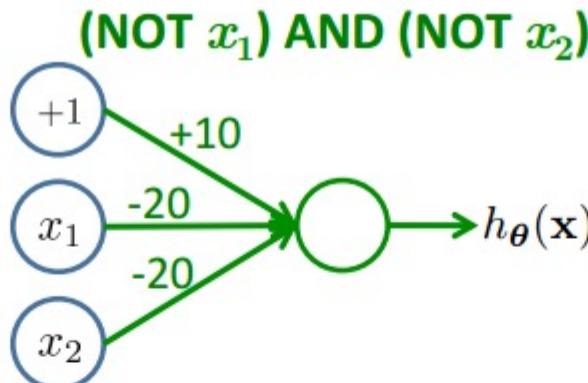
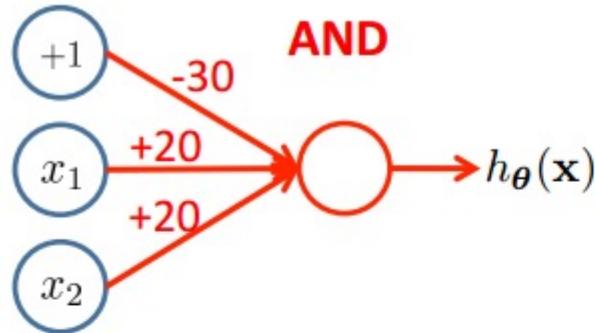


x_1	x_2	$h_{\theta}(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

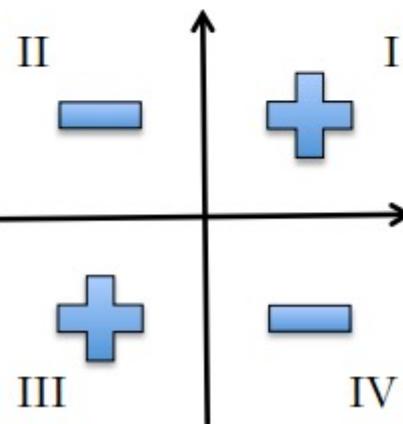




Combining Representations to Create Non-Linear Functions

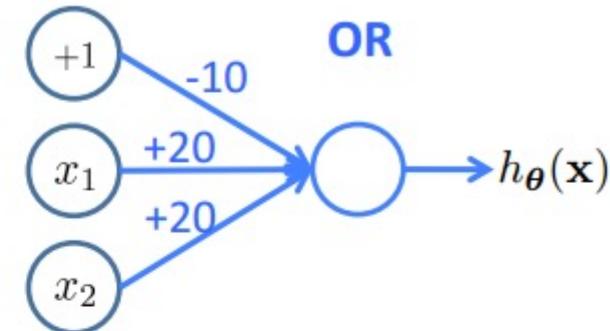
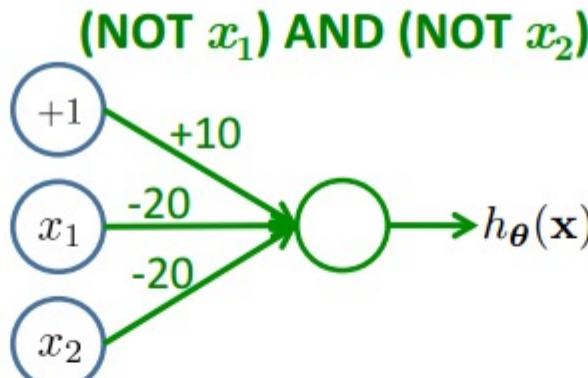
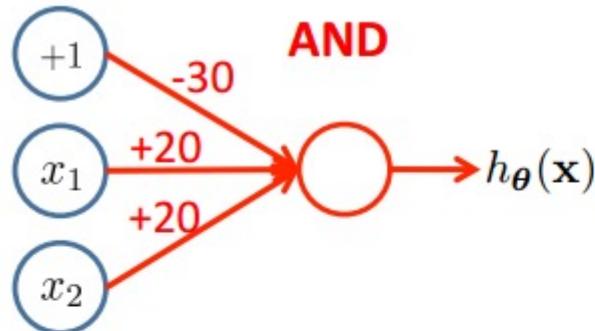


XNOR

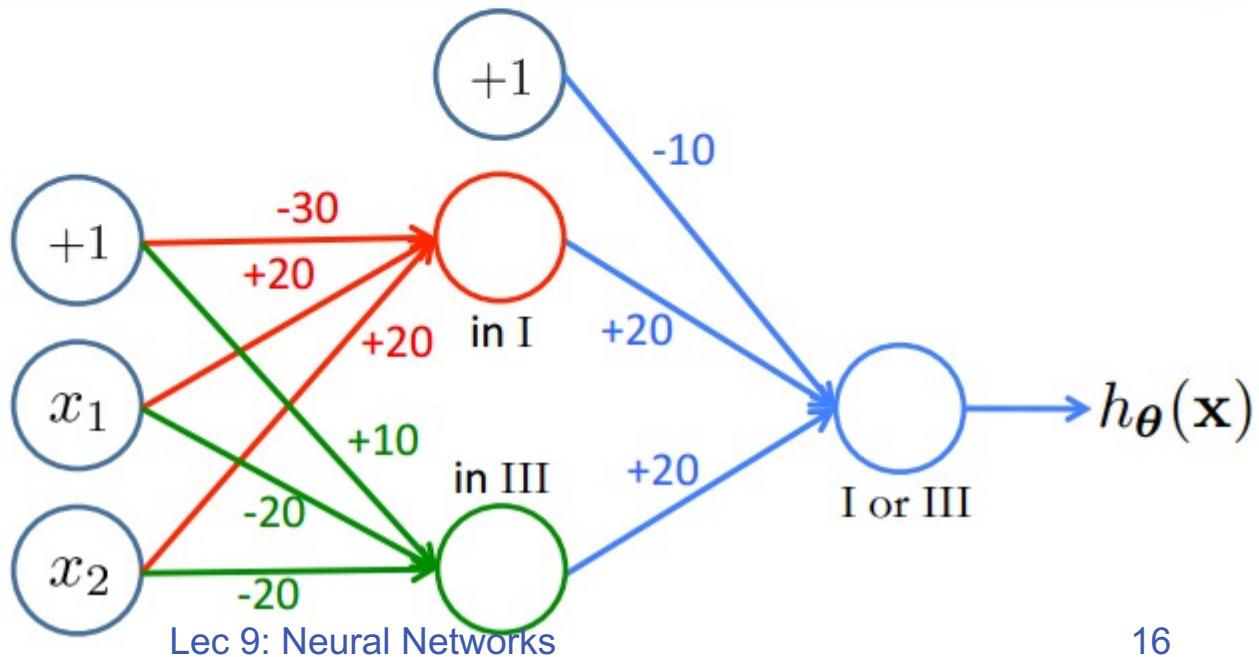
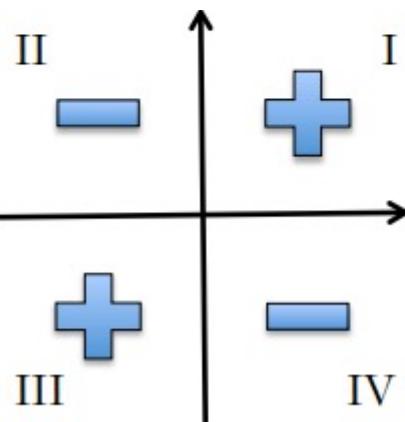


XNOR

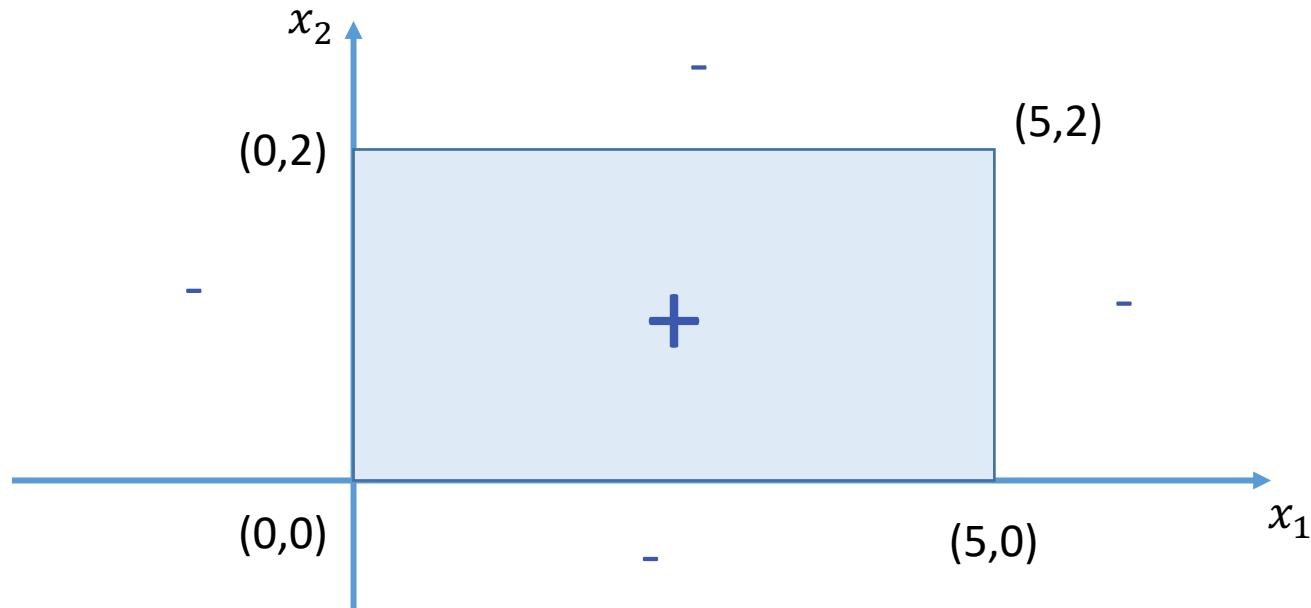
Combining Representations to Create Non-Linear Functions



XNOR



Exercise



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

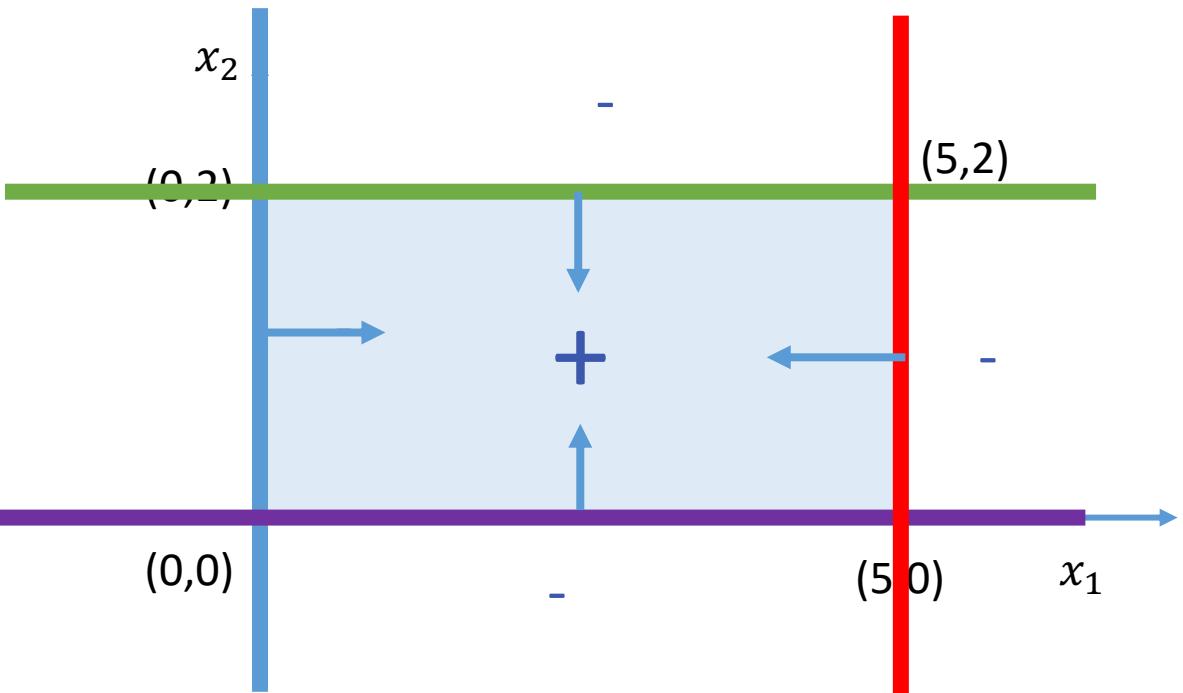
$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

If all the samples inside the rectangle are positive;
otherwise are negative

Show a feedforward NN can classify all the samples correctly
For simplicity, we assume $g(z)$ is a step function.
What are $\Theta^{(1)}$ and $\Theta^{(2)}$

Exercise



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

If all the samples inside the rectangle are positive;
otherwise are negative

Show a feedforward NN can classify all the samples correctly
For simplicity, we assume $g(z)$ is a step function.
What are $\Theta^{(1)}$ and $\Theta^{(2)}$

$$\begin{cases} x_1 & > 0 \\ 5 - x_1 & > 0 \\ x_2 & > 0 \\ -x_2 & > 0 \end{cases}$$

$$\Rightarrow \Theta^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ 5 & -1 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & -1 \end{bmatrix}$$

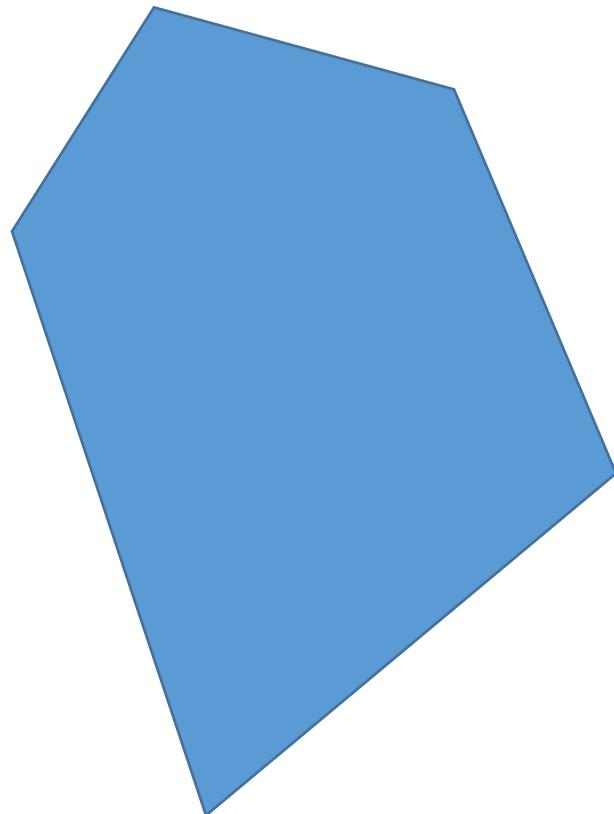
$$\Theta_0^{(1)} \Theta_1^{(1)} \Theta_2^{(1)}$$

$$-3.5 + a_1 + a_2 + a_3 + a_4 > 0$$

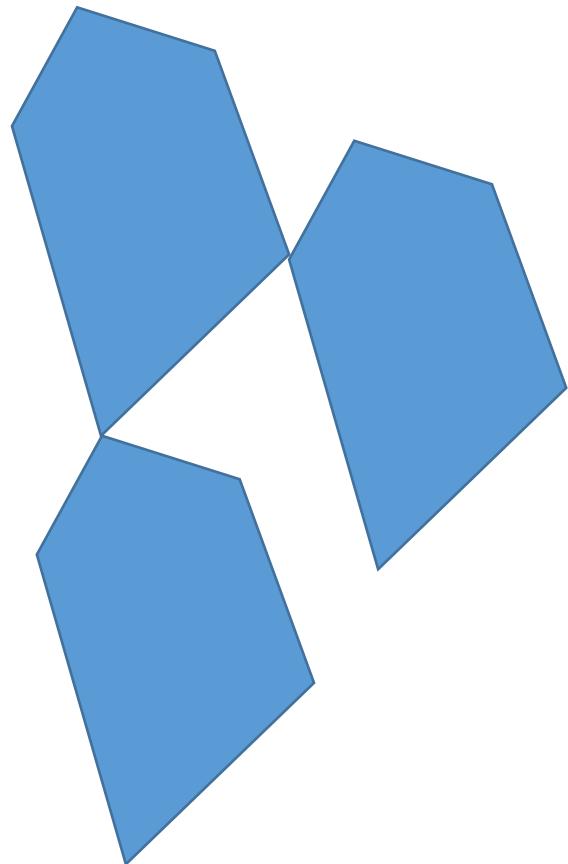
$$\Theta^{(2)} = [-3.5 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$\Theta_0^{(2)} \Theta_1^{(2)} \Theta_2^{(2)} \Theta_3^{(2)} \Theta_4^{(2)}$$

Arbitrary Decision Boundary

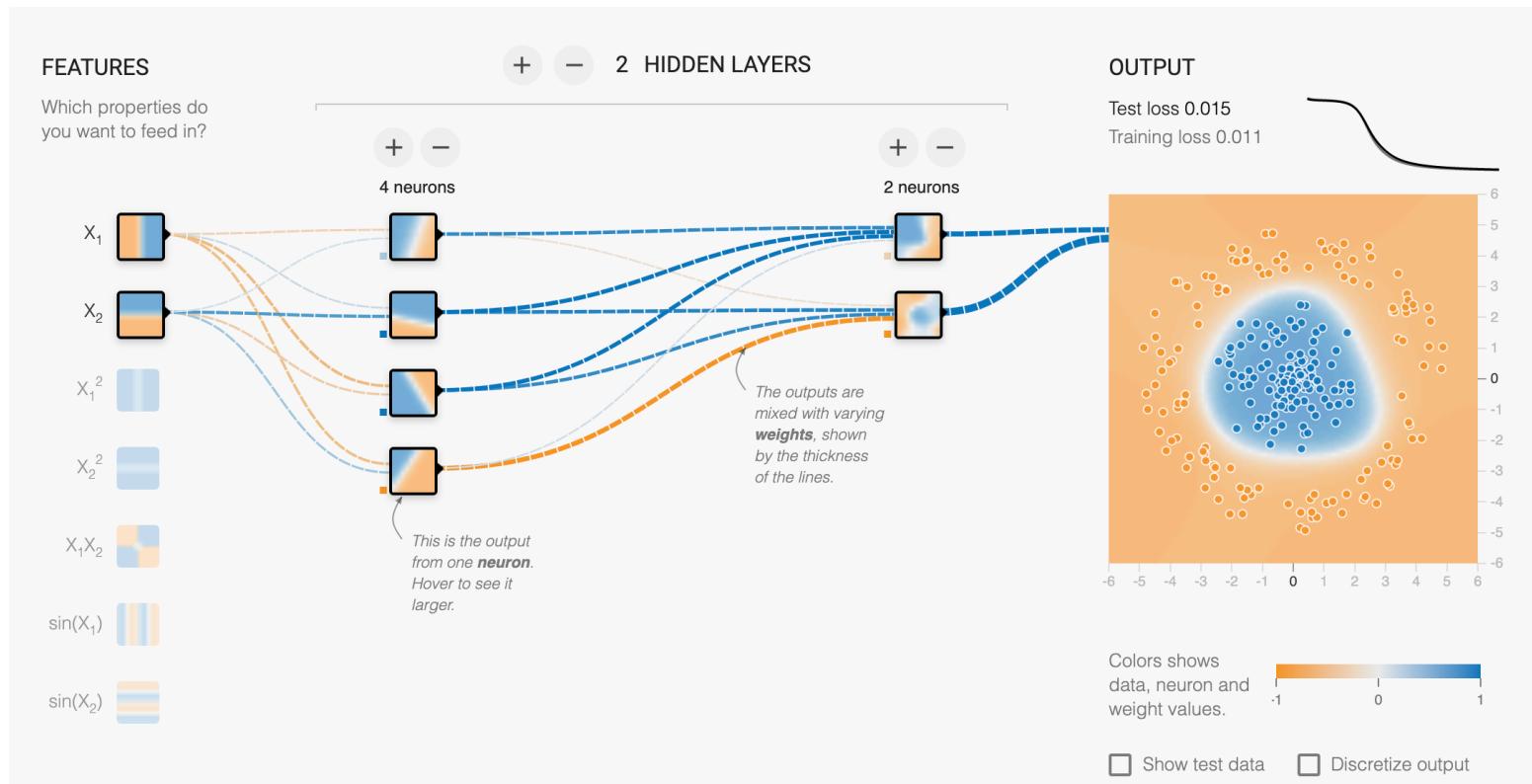


Arbitrary Decision Boundary



Neural Network Training Animation

❖ <https://playground.tensorflow.org/>



Neural Network Learning

Maximum Likelihood

- ❖ Training data: $S = \{(x_i, y_i)\}$, m examples
 $y_i \in \{0, 1\}$
- ❖ Consider a NN $h_\Theta(x) \in [0, 1]$ modeling $P(y = 1|x)$

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$

Remember in logistic regression, $h_{w,b}(x) = \sigma(w^T x + b)$

If we choose the activation function g as a sigmoid function,
this part is equivalent to a logistic regression with input a

Maximum Likelihood

- ❖ Training data: $S = \{(x_i, y_i)\}$, m examples
 $y_i = \{0,1\}$
- ❖ Consider a NN $h_\Theta(x) \in [0,1]$ modeling $P(y = 1|x)$
- ❖ Maximum Likelihood estimator $\Theta^* = \arg \max_{\Theta} L(\Theta; S)$
 $L(\Theta; S) = \prod_{i=1}^m P_\Theta(y_i|x_i)$

$$P_\Theta(y_i|x_i) = \begin{cases} h_\Theta(x_i), & y_i = 1 \\ 1 - h_\Theta(x_i), & y_i = 0 \end{cases}$$

- ❖ We can rewrite $L(\Theta; S)$ as

$$L(\Theta; S) = \prod_{i=1}^m h_\Theta(x_i)^{y_i} (1 - h_\Theta(x_i))^{1-y_i}$$

Minimum Negative Log-Likelihood & Cross-Entropy Loss

- ❖ Training data: $S = \{(x_i, y_i)\}$, m examples

$$y_i = \{0,1\}$$

- ❖ Consider a NN $h_\Theta(x) \in [0,1]$ modeling $P(y = 1|x)$

- ❖ Likelihood $L(\Theta; S)$ is

$$L(\Theta; S) = \prod_{i=1}^m h_\Theta(x_i)^{y_i} (1 - h_\Theta(x_i))^{1-y_i}$$

- ❖ Log-Likelihood:

$$\log L(\Theta; S) = \sum_i^m [y_i \log h_\Theta(x_i) + (1 - y_i) \log(1 - h_\Theta(x_i))]$$

- ❖ Optimal Θ can be obtained by solving $\arg \min_{\Theta} J(\Theta)$

$$J(\Theta) = - \sum_i^m [y_i \log h_\Theta(x_i) + (1 - y_i) \log(1 - h_\Theta(x_i))]$$

Cross-entropy between prediction $h_\Theta(x)$ and true label y_i

Lec 9: Neural Networks

Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $\Theta \leftarrow \mathbf{0} \in \mathbb{R}^n$

2. For epoch 1 ... T :

3. For (x, y) in \mathcal{D} :

4. Update $\Theta \leftarrow \Theta - \eta \nabla J(\Theta)$

5. Return Θ

(Similar to logistic regression)

$$J(\Theta) = -\sum_i^m [y_i \log h_\Theta(x_i) + (1 - y_i) \log(1 - h_\Theta(x_i))]$$

Optimizing the Neural Network

$$J(\Theta) = -\sum_i^m [y_i \log h_\Theta(x_i) + (1 - y_i) \log(1 - h_\Theta(x_i))]$$

- ❖ Need to compute $\nabla J(\Theta)$

Chain Rule

- ❖ Given a function

$$f(x) = A(B(C(x)))$$

- ❖ The derivative is

$$f'(x) = A'(B) \cdot B'(C) \cdot C'(x)$$

Backpropagation through Computation Graphs

Computation Graphs and Backpropagation

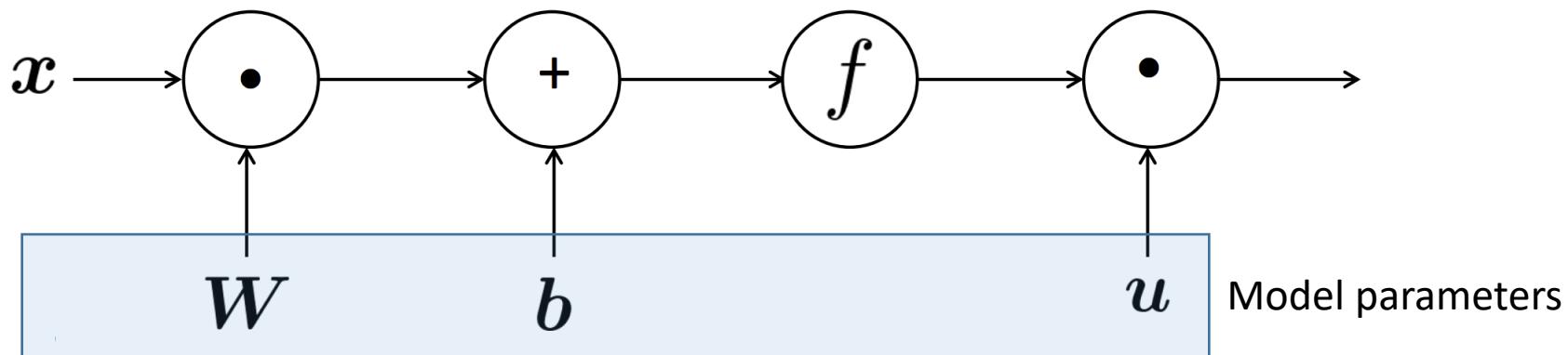
- ❖ Consider the NN on the right
- ❖ We represent NN as a graph

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

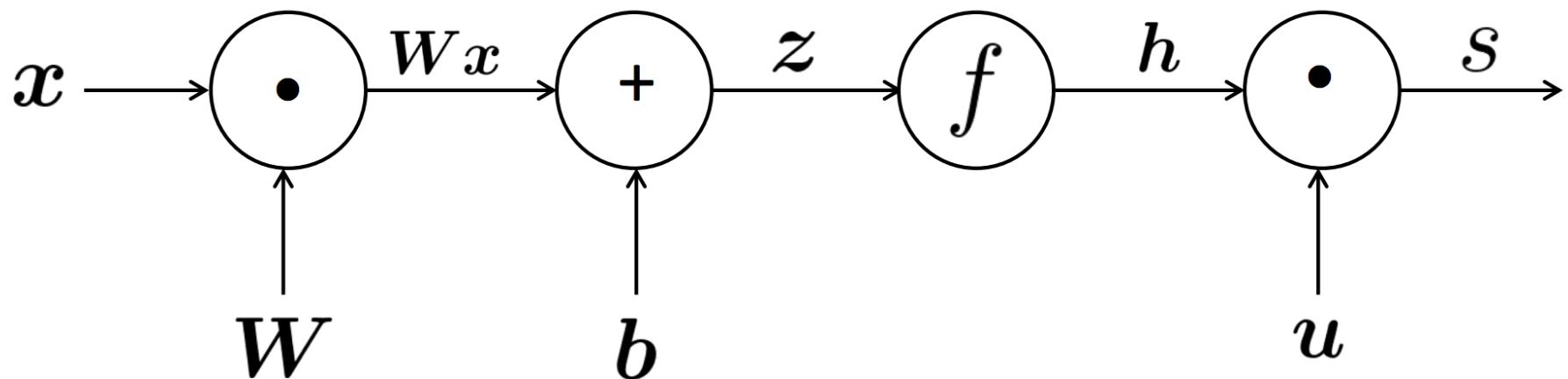
$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)



Forward Propagation

- ❖ Edges pass along result of the operation



$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

Back Propagation

$$s = \mathbf{u}^T \mathbf{h}$$

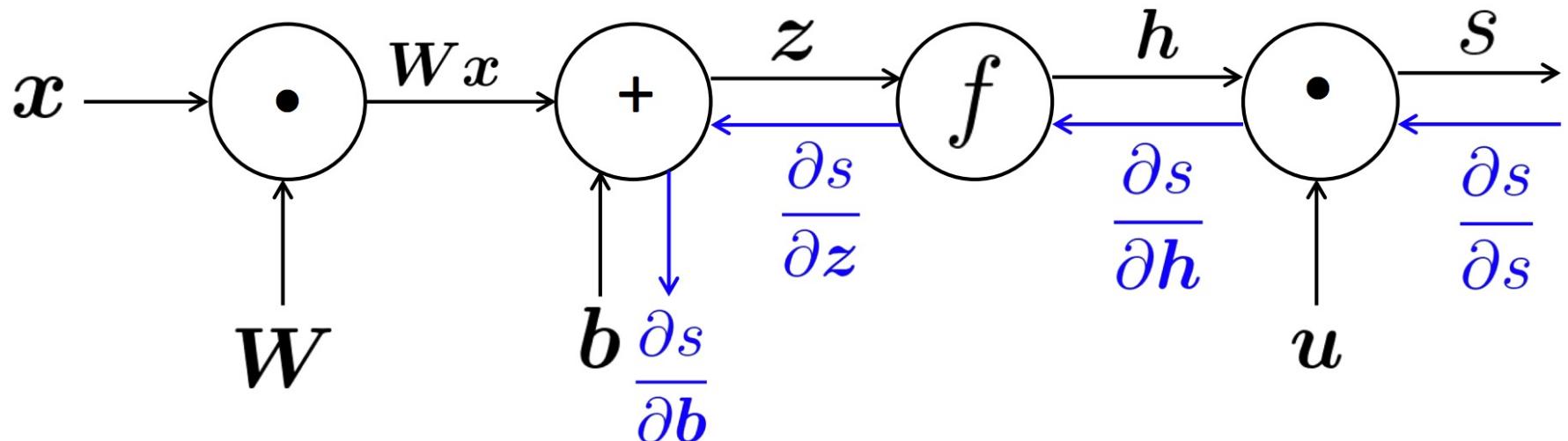
$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

❖ Compute $\frac{\partial s}{\partial b}$

$$\text{Chain Rule: } \frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \dots$$



Back Propagation

$$s = \mathbf{u}^T \mathbf{h}$$

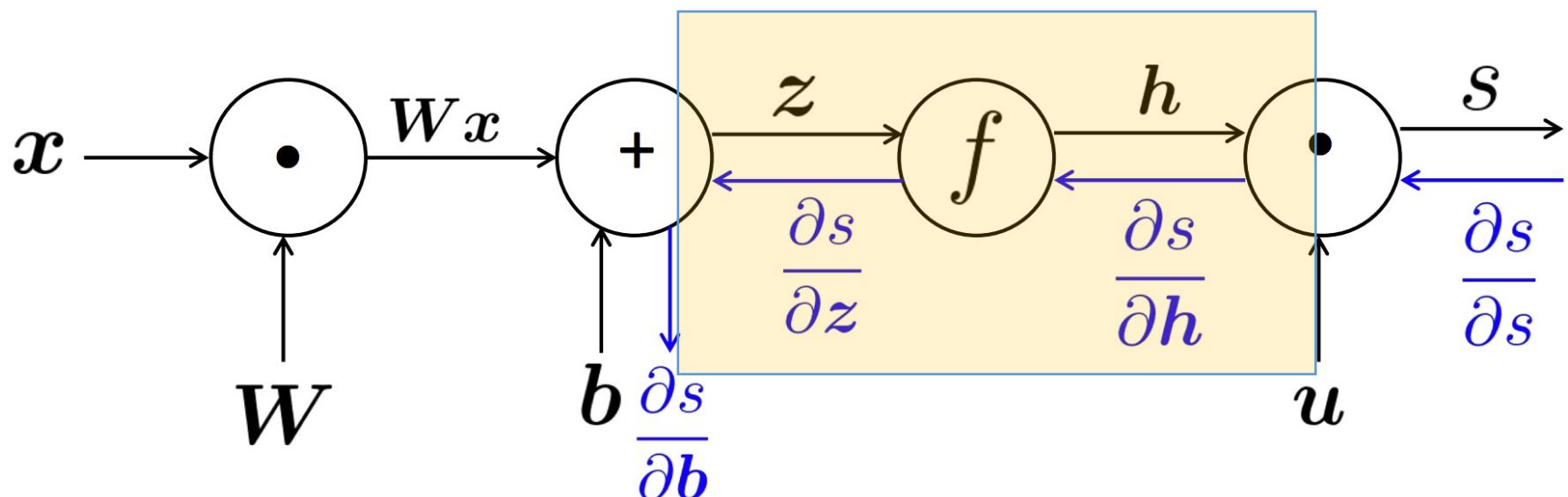
$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

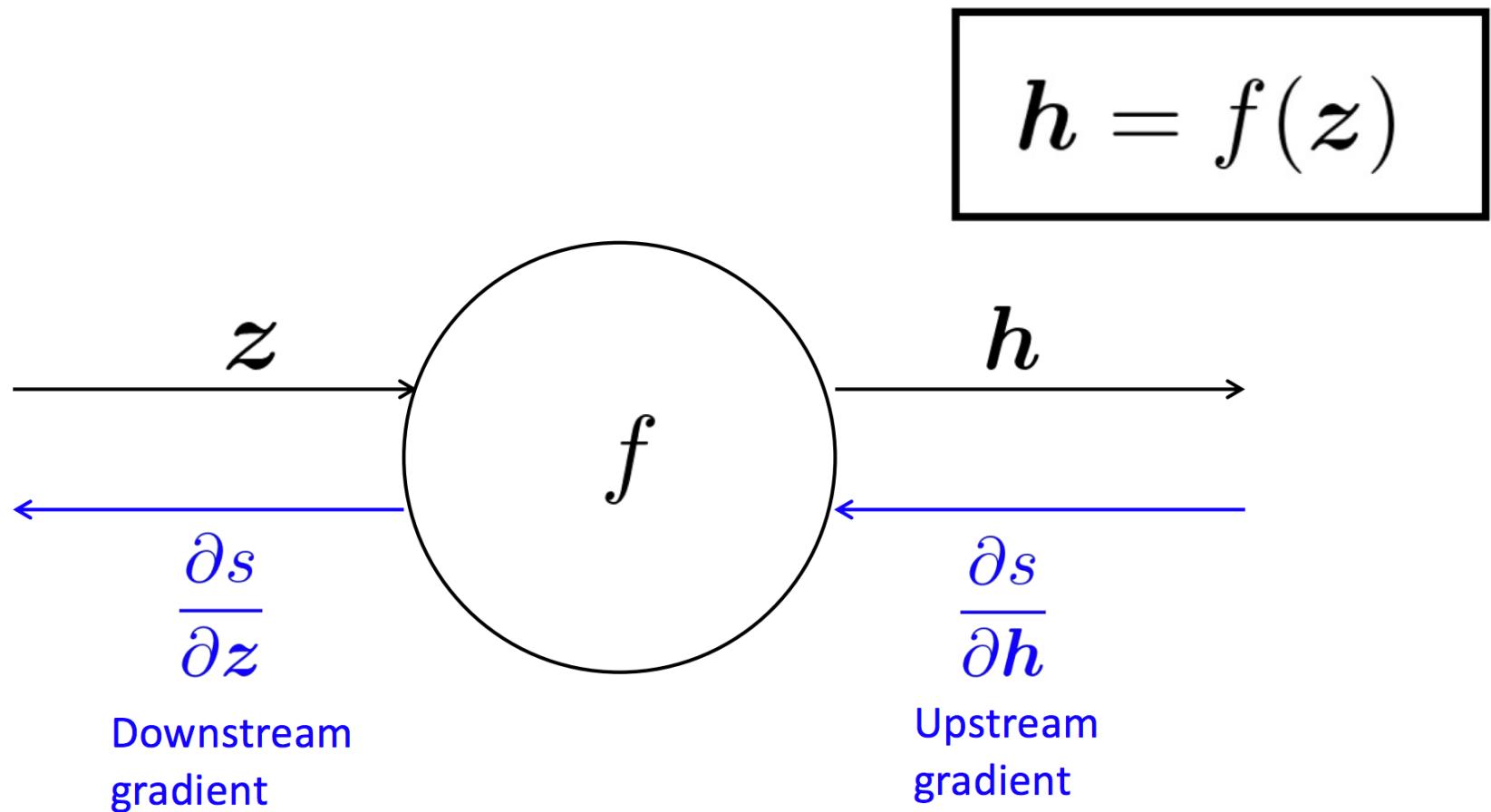
\mathbf{x} (input)

❖ Compute $\frac{\partial s}{\partial b}$

$$\text{Chain Rule: } \frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \frac{\partial s}{\partial h} \frac{\partial h}{\partial z} \frac{\partial z}{\partial b}$$

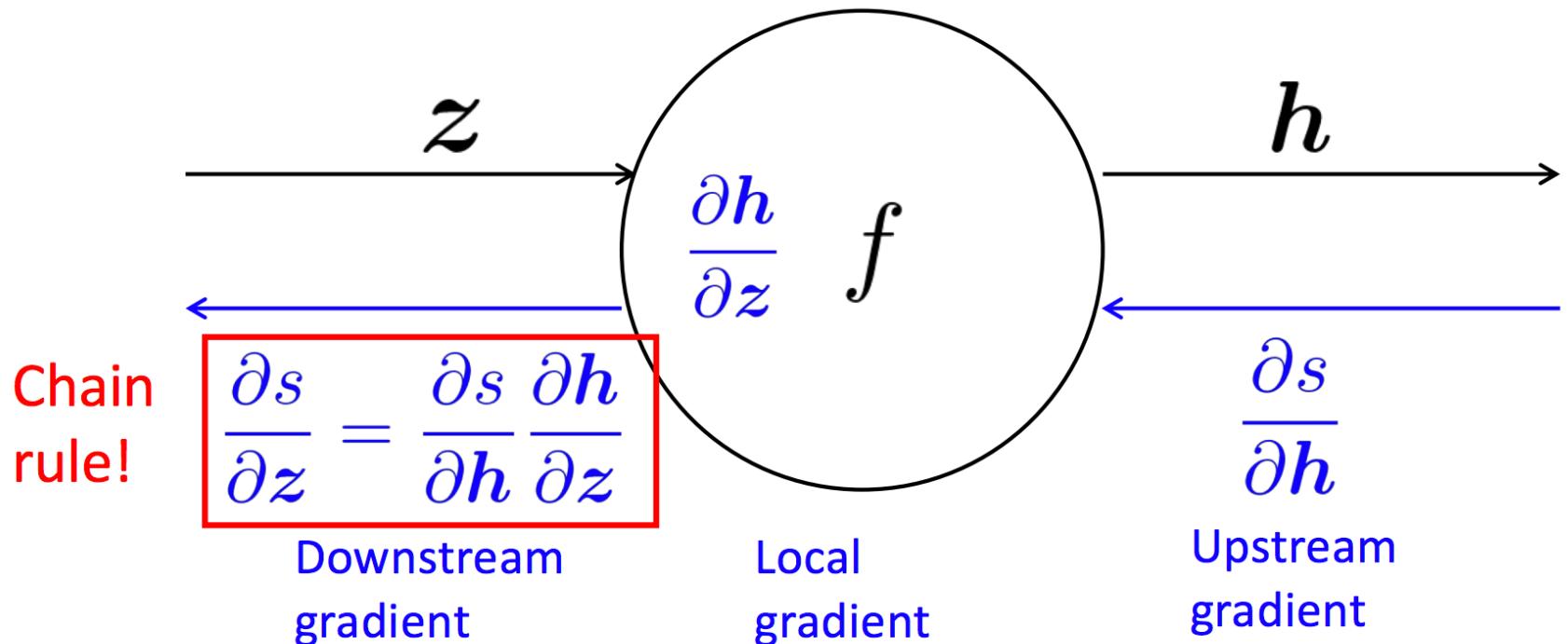


Backpropagation: Single Node



Chain Rule

$$h = f(z)$$



Back Propagation

$$s = \mathbf{u}^T \mathbf{h}$$

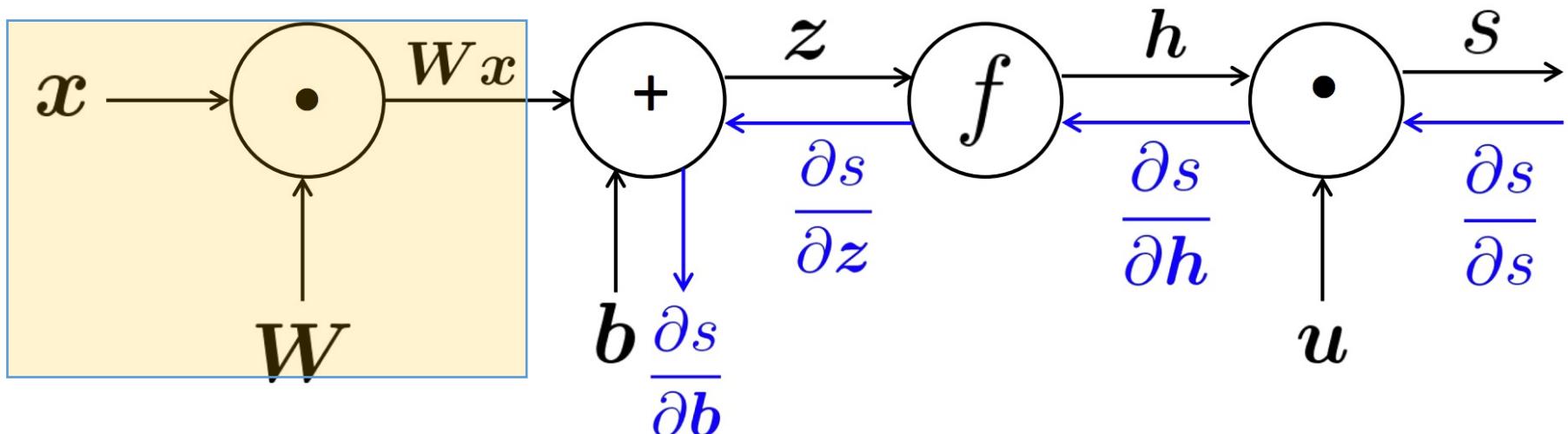
$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

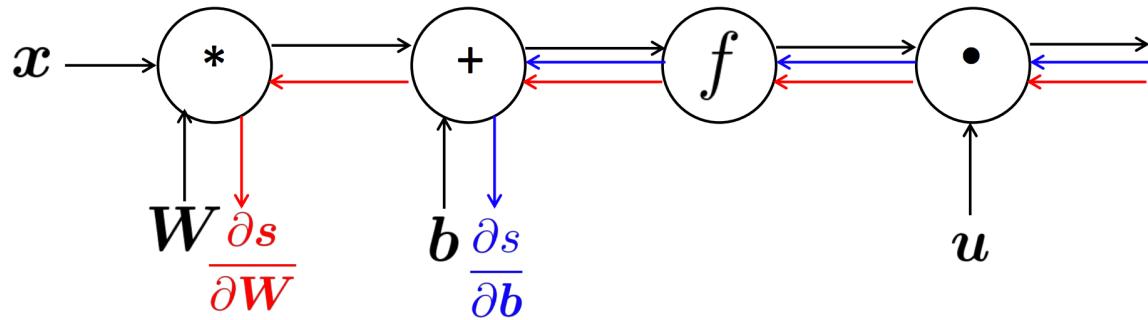
\mathbf{x} (input)

❖ Compute $\frac{\partial s}{\partial b}$

$$\text{Chain Rule: } \frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \dots$$



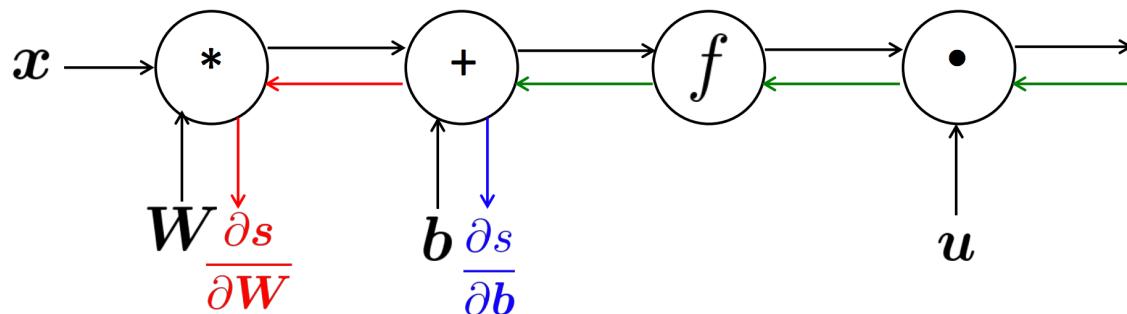
Compute all gradients at once



Naïve way to compute gradients:

Compute each component separately

⇒ Redundant computation



Example

$$f(x, y, z) = (x + y) \max(y, z)$$

$$x = 1, y = 2, z = 0$$

Draw the computation graph and calculate $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Example

$$f(x, y, z) = (x + y) \max(y, z)$$

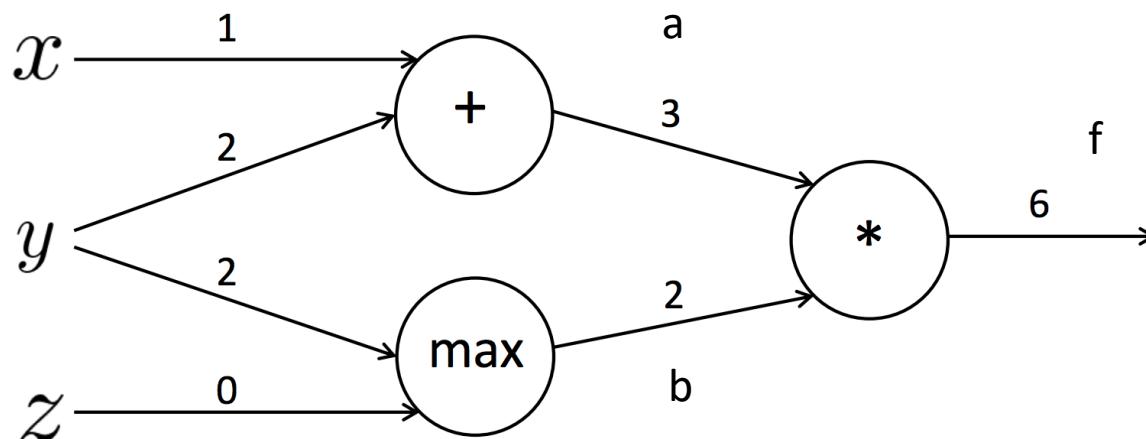
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



Example

$$f(x, y, z) = (x + y) \max(y, z)$$

$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

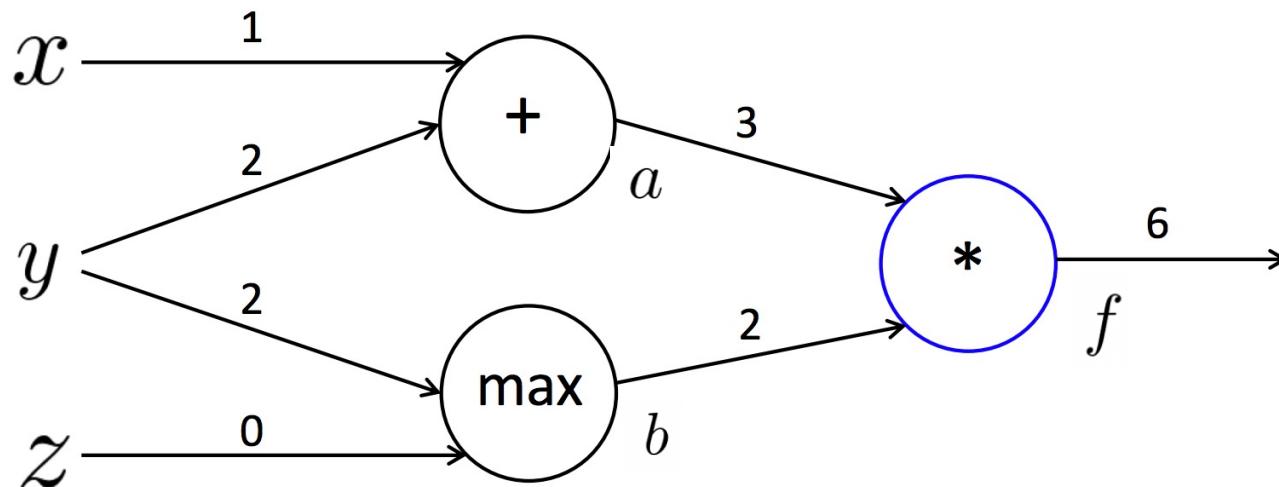
$$f = ab$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



Example

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

$$f(x, y, z) = (x + y) \max(y, z)$$

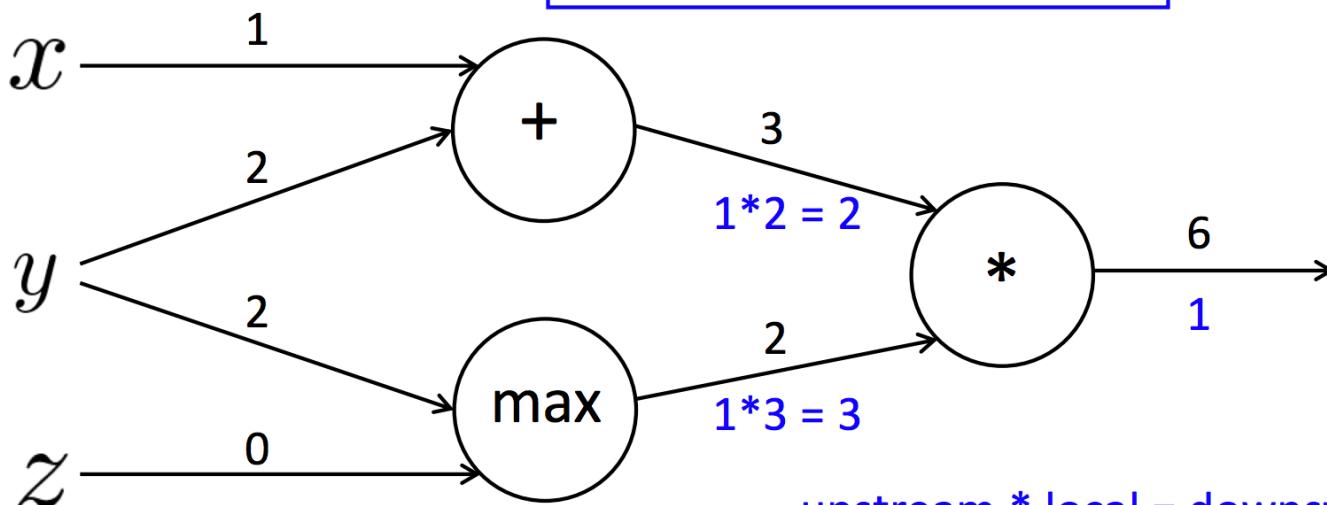
$$x = 1, y = 2, z = 0$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



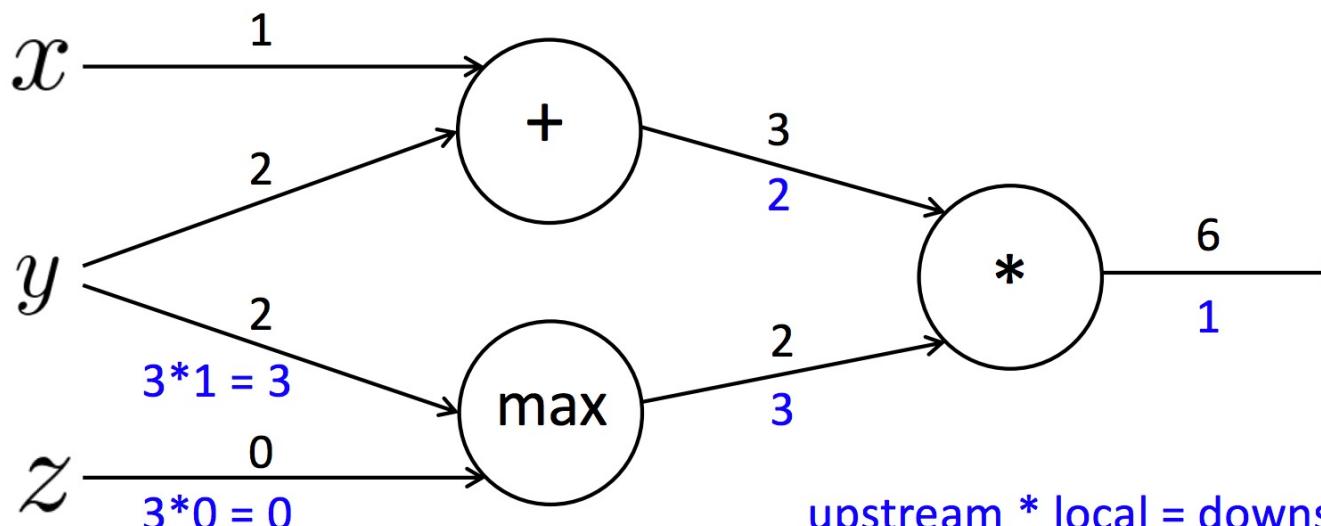
Example

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



$$f(x, y, z) = (x + y) \max(y, z)$$

$$x = 1, y = 2, z = 0$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$

Example

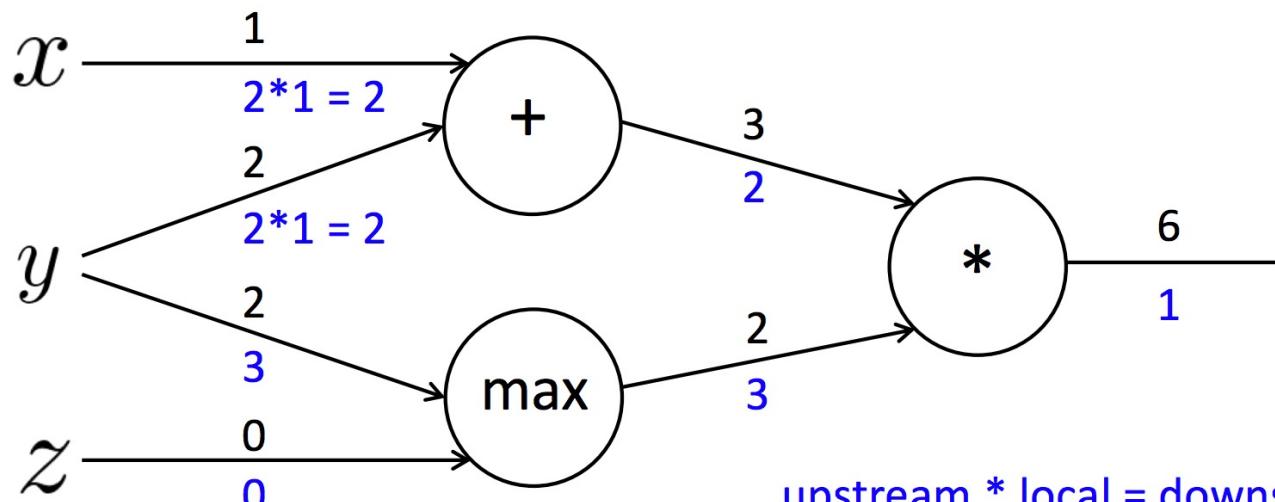
$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$

upstream * local = downstream

Example

$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

$$\frac{\partial f}{\partial x} = 2$$

$$\frac{\partial f}{\partial y} = 3 + 2 = 5$$

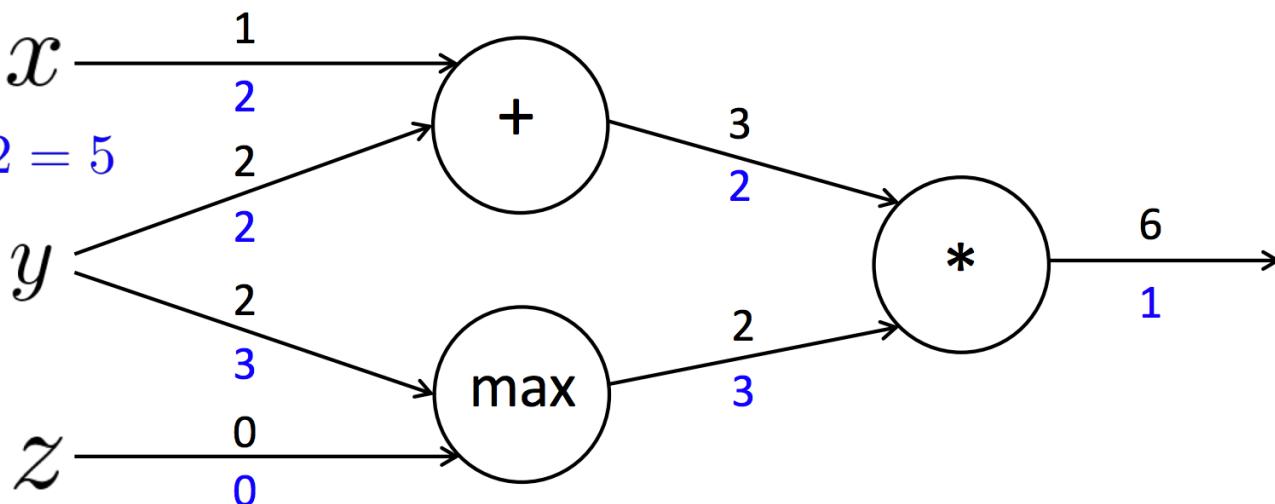
$$\frac{\partial f}{\partial z} = 0$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$

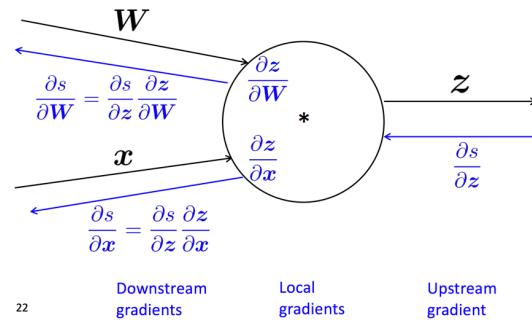


Why you should understand Backprop

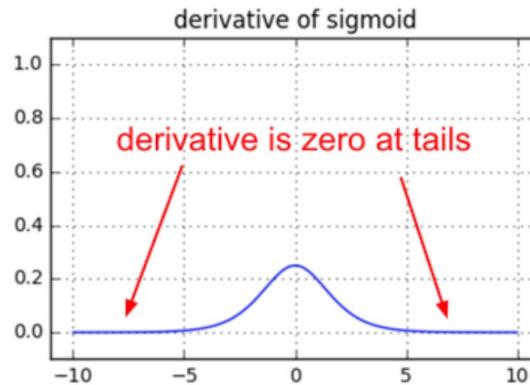
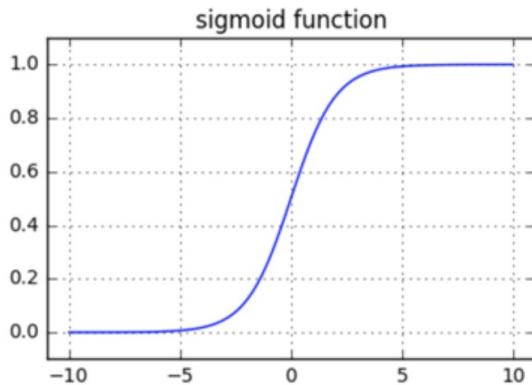
- ❖ Modern deep learning library implements backprop as a black-box for you
 - ❖ You can take a plane without knowing why it flies
 - ❖ but you're designing aircraft...
- ❖ Backpropagation doesn't always work perfectly.
 - ❖ Understanding why is crucial for debugging and improving models

<https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>

Example: Gradient of sigmoid



```
z = 1/(1 + np.exp(-np.dot(W, x))) # forward pass  
dx = np.dot(W.T, z*(1-z)) # backward pass: local gradient for x  
dW = np.outer(z*(1-z), x) # backward pass: local gradient for W
```



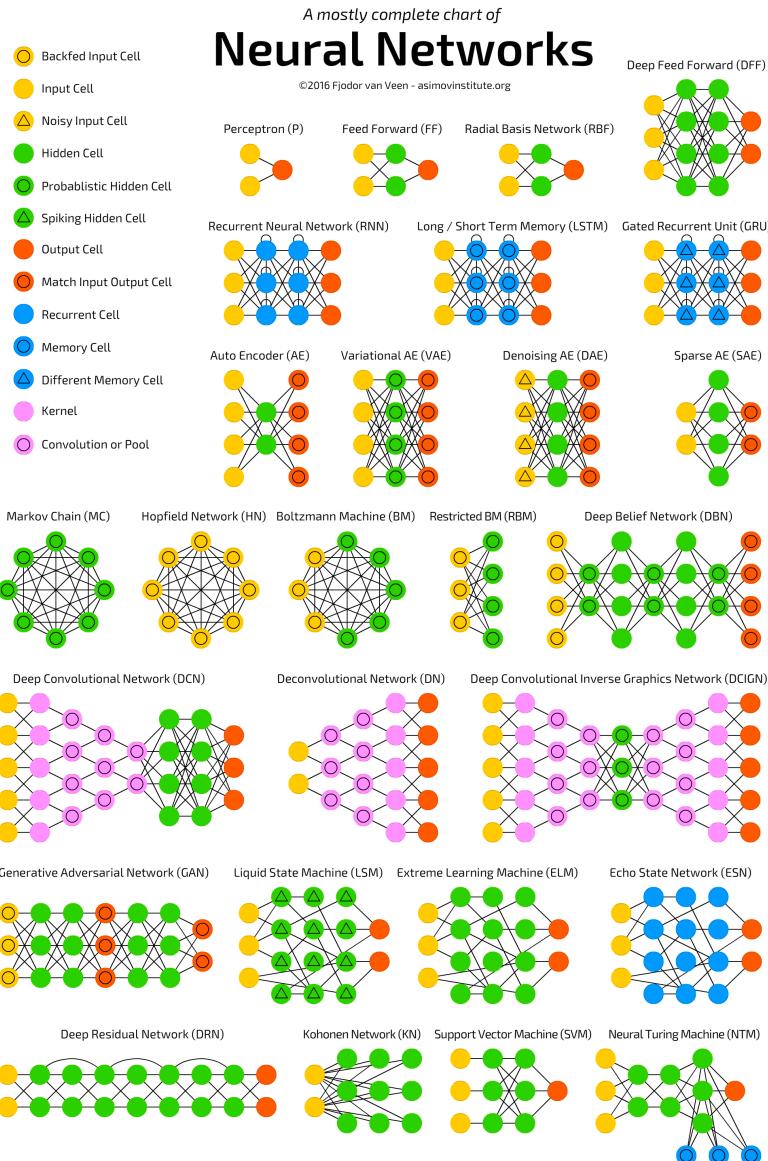
vanish gradient issue

More Details

- ❖ Parameter Initialization
 - ❖ Normally initialize weights to small random values; various designs
- ❖ Optimizer
 - ❖ Usually SGD works
 - ❖ Several SGD variants (e.g., ADAM) automatically adjust learning rate based on an accumulated gradient

A neural network zoo

- ❖ The flexibility of NN allows us to try out different ideas
- ❖ However, there is no magic



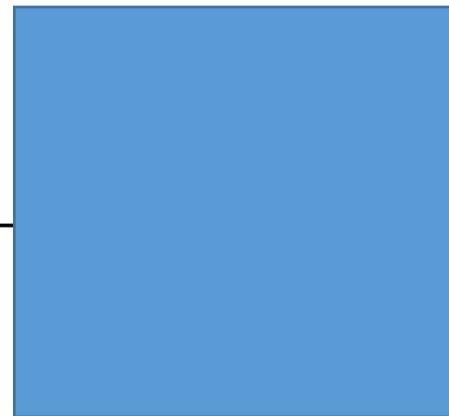
Modeling with Neural Networks

(Advanced Topic/Not Included in Final)

Example – Language Model

- ❖ Predict next word

the students opened their _____



Idea 1: A fixed-window neural Language Model

output distribution

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

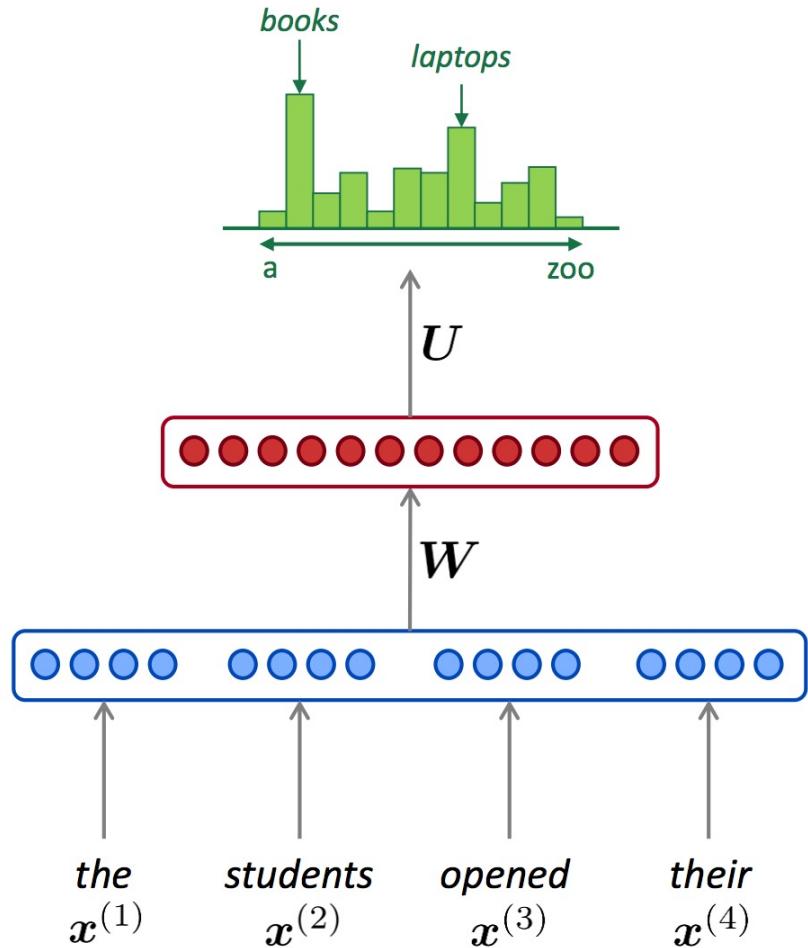
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

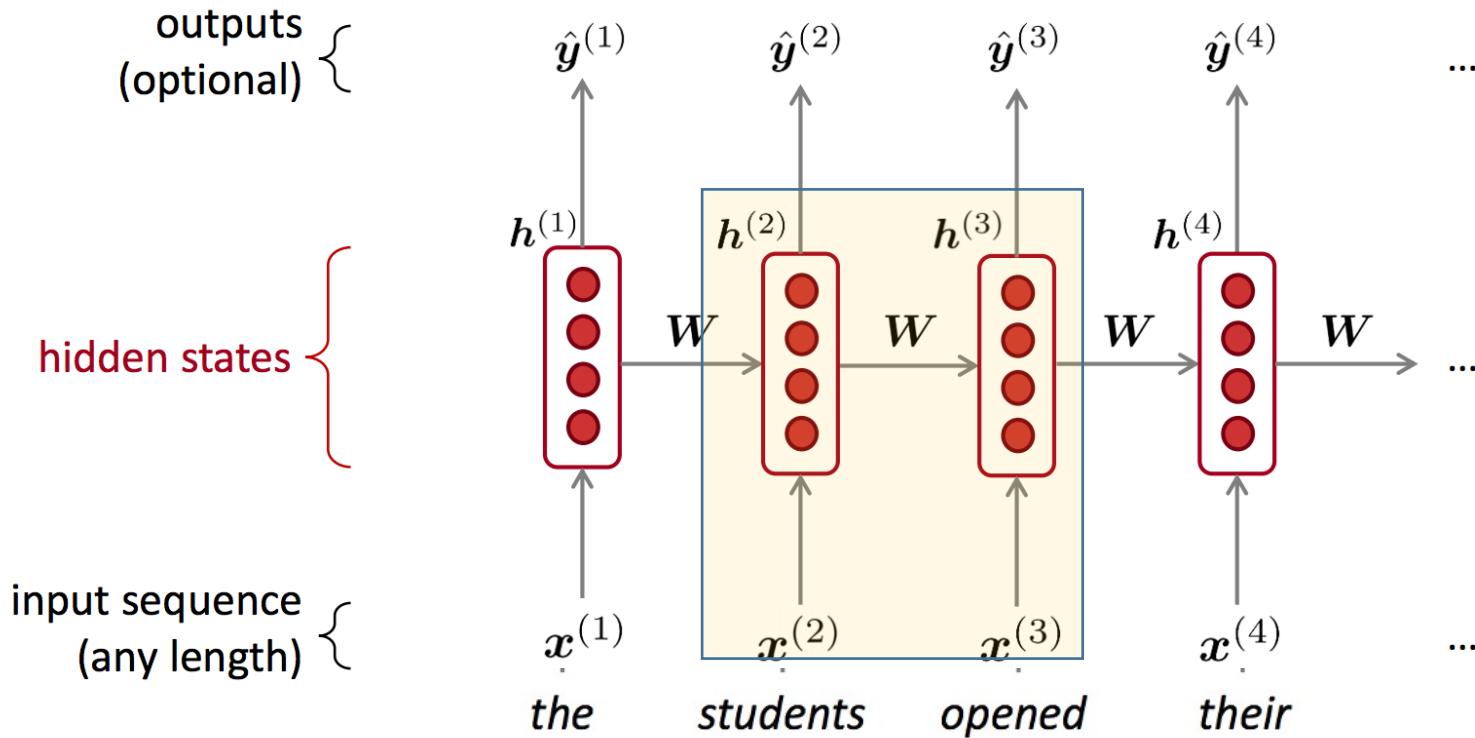
$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$

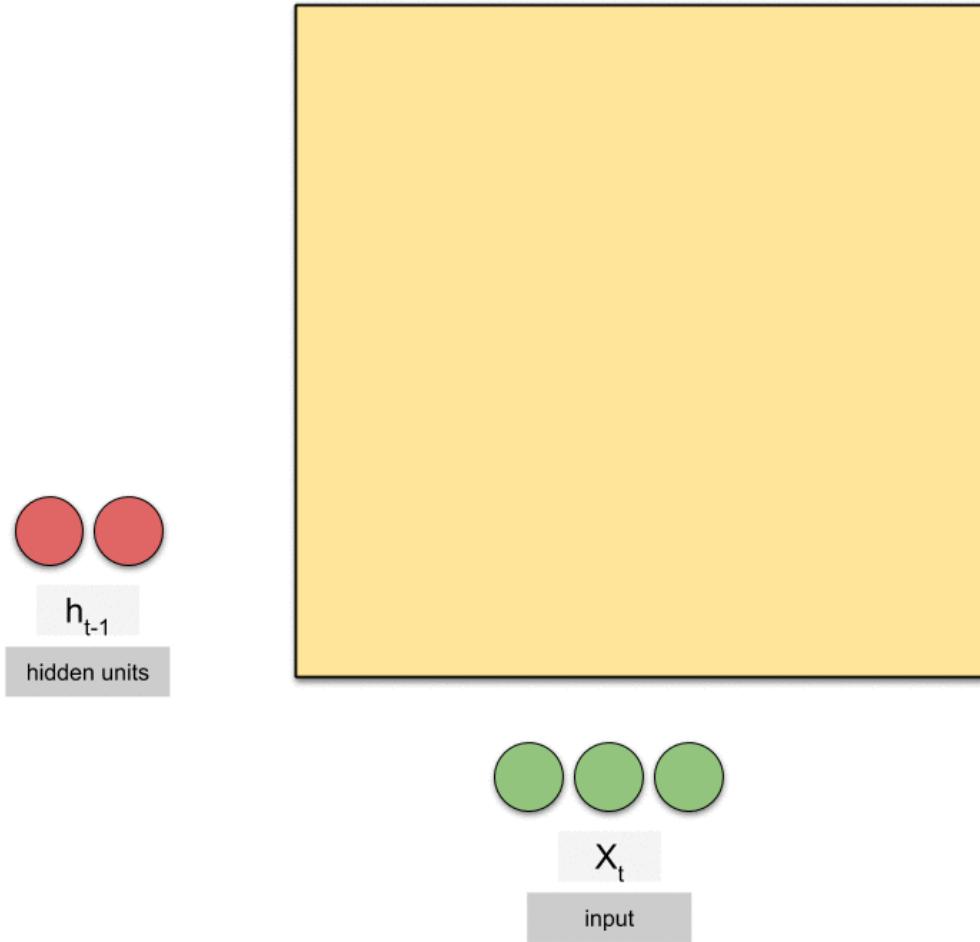


Idea 2: Recurrent Neural Networks (RNN)



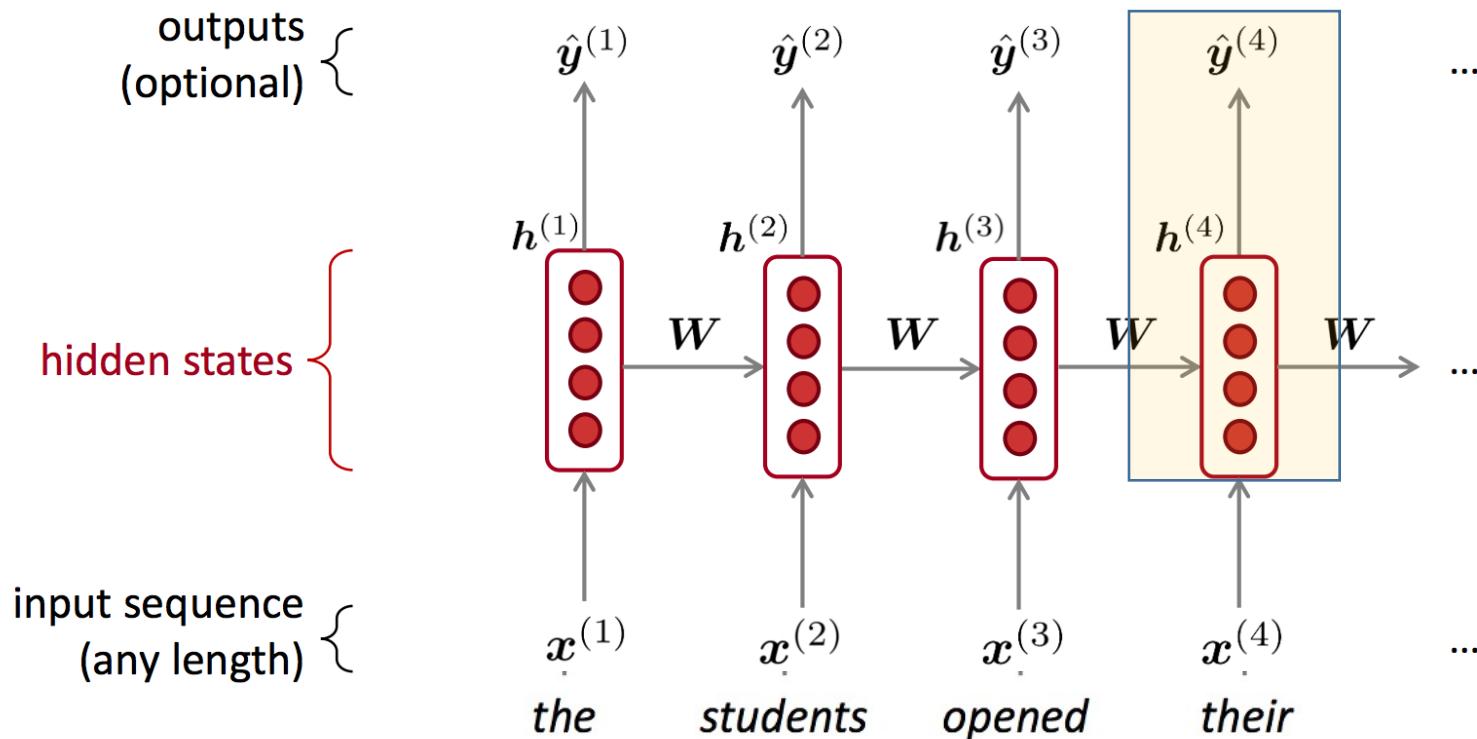
Core idea: Apply the same weights W repeatedly

Recurrent Neural Network



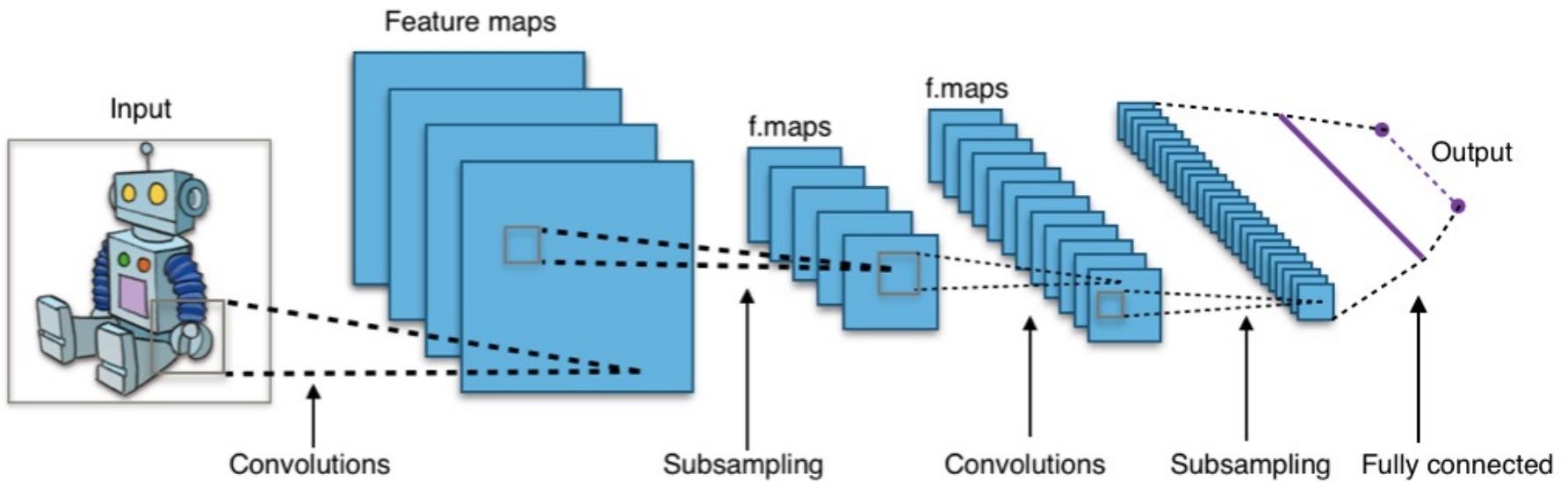
<https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45>

Prediction using Latent State

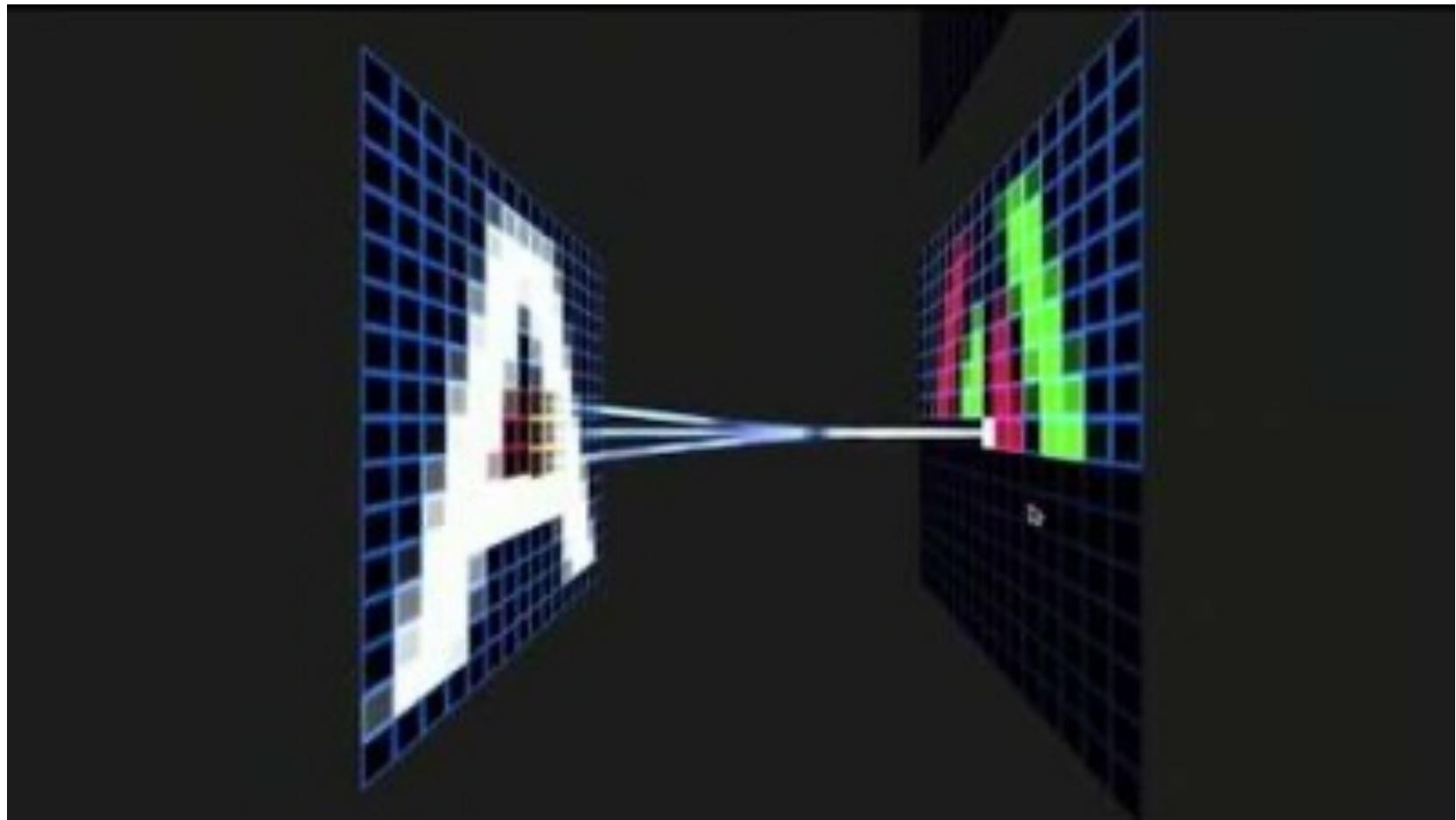


Core idea: Apply the same weights W repeatedly

Idea 3: Convolutional NN



Convolutional NN



<https://www.youtube.com/watch?v=f0t-OCG79-U>